Critic2 user's guide

Author: Alberto Otero-de-la-Roza, Ángel Martín Pendás, Víctor Luaña.
Contact: alberto@fluor.quimica.uniovi.es
Address: Departamento de Química Física y Analítica,
Universidad de Oviedo, 33007 Oviedo, Spain.

Contents

1 Introduction

Critic2 is a program for the analysis and representation of solid-state electron densities and other scalar fields. This program provides an interface to many solid-state electronic structure programs and makes it possible to analyze, manipulate and interconvert scalar fields (electron densities and others) and crystal structures.

The types of analyses critic2 is able to perform are based on Bader's Quantum Theory of Atoms in Molecules (QTAIM). Critic2 encapsulates common methods for topological analysis under periodic boundary conditions and applies them, irrespective of the source of the electron density. This primarily includes locating the complete set of critical points (topology) and integrating atomic regions (basins). Several specialized algorithm for the latter are available.

Despite its origins, critic2 is not limited to QTAIM analysis. Many other convenient features are available such as sundry techniques related to QTAIM (e.g. Hirshfeld charges, NCIplot,...), tools for graphical representations and crystallographic computation, etc. At the most basic level, critic2 is also useful to perform arithmetic manipulations on fields defined in real space and to interconvert between crystal structure and field file formats.

Many scalar fields other than the electron density are amenable to an analysis using the same tools as in QTAIM: finding the topology and integrating the attractor basins. These techniques (called Quantum Chemical Topology, QCT, by P. Popelier) can be applied to a variety of scalar fields including the ELF and the Laplacian of the electron density. Critic2 can, in principle, perform all these tasks for any field provided by the user or calculated within the code.

Critic2 is described in Comput. Phys. Commun. 180 (2009), 157-166 and Comput. Phys. Commun. 185 (2014), 1007-1018. Please, cite these references if you find this program useful. The file THANKS contains a complete list of contributors and other programs from where parts of critic2 have been adapted.

2 Command-line options, usage, and input overview

Critic2 accepts a single input file (usually but not necessarily with extension .incritic). The command-line syntax is:

```
$ critic2 [-q] [-h] [-r /path/to/critic2] ...
          [file.incritic [file.outcritic]]
```

If both input and output files are present, the output is redirected to file.outcritic. Otherwise, it is written to standard output. If no file.incritic is present, the input is taken from standard input. The extensions (incritic and outcritic) are not mandatory. It is also valid to use:

```
$ critic2 < file.incritic > file.outcritic
```

Alternatively, you can use the standard input and standard output:

```
$ critic2
```

and then enter the commands by hand. In addition, depending on the tasks in the incritic file, auxiliary files may be generated, usually with the same root as file.incritic ("file", in the example above) or "stdin" if no input file is given.

The -h option ("help") prints a short help message and exits. The -q option ("quiet") inhibits the initial and final messages. This is useful when using critic2 in a pipe command. The -r option tells critic2 where to find its data files. If /path/to/critic2 is given, the data files (wfc, dic, etc.) should be in /path/to/critic2/dat.

The input file usually contains a specification of the crystal structure using the CRYSTAL keyword. For instance:

```
CRYSTAL file.struct
```

After the crystal structure is read, one or more fields are loaded using the keyword LOAD:

```
LOAD file.clmsum file.struct
```

Unless otherwise stated in this manual, the units in critic2 are **atomic units**. In particular, distances are in bohr and the electron density is in electrons/bohr ^{3}. After loading the relevant fields, keywords are used to perform the required tasks. For instance, AUTO (automatic determination of critical points), INTEGRALS (atomic integration), NCIPLOT (non-covalent interaction index plots), etc.

A concise description of the critic2 keyword syntax is given in the syntax.txt file. If you are already familiar with this program, that file will probably be more useful to you than this guide.

3 Input format, output format, and notation

The input for critic2 is free-format and case-insensitive. Lines preceded by # are treated as comments. The input syntax is keyword-oriented. The first word in a (non-blank) input line is the "main" keyword, and determines one task that critic2 is required to perform. In the remainder of this manual, these main keywords and their usage are described. Keywords are written in CAPS (and sometimes in color, if they are within a code block). Input variables are denoted using a suffix to indicate their type: a real number (.r), an integer (.i) or a string (.s). Almost anywhere that a number is expected, it is possible to use an arithmetic expression (see the Arithmetic expressions, variables, and functions in critic2 section). If an arithmetic expression is required, quotes ("expr.s") are used. When several alternative keywords are possible, the or symbol (|) is used. Square brackets ([]) denote optional keywords and curly braces ({}) are used for grouping.

Some of the sections in the rest of this manual contain an "additional options" subsection. These provide some independent keywords that control the behavior of critic2 and are meant to be used either before or after the keywords in the section in which they appear. For instance, NOGUESS can be used before CRYSTAL to deactivate the automatic calculation of the crystal symmetry:

```
NOGUESS
CRYSTAL benzene.cif
```

Hence, NOGUESS appears in this manual as an additional option to CRYSTAL.

The critic2 output is mostly self-explanatory, although there are a number of key concepts that need to be understood. The crystal structure in critic2 is represented by two lists of atoms. The **non-equivalent atom list** contains the atoms in the asymmetric unit, that is, the minimal list of atoms that generate all the atomic positions in the crystal by symmetry. The **cell atom list**, equivalently called the **complete atom list**, contains all atoms in the current unit cell. The non-equivalent atom list reproduces the complete atom list by applying all symmetry operations (except lattice translations) known to critic2.

For instance, an input for the conventional cell of the fluorite (CaF2) crystal is:

```
CRYSTAL
 SPG f m -3 m
 CELL 5.463 5.463 5.463 90 90 90 ANG
 NEQ 0 0 0 ca
 NEQ 1/4 1/4 1/4 f
ENDCRYSTAL
```

The non-equivalent atom list contains Ca at (0 0 0) with multiplicity 4 and F at (1/4 1/4 1/4) with multiplicity 8. The cell atom list contains 4 Ca atoms at (0 0 0), (1/2 1/2 0), etc. and 8 F atoms at (1/4 1/4 1/4), (3/4 1/4 1/4), etc. Note that critic2 uses the input cell as given; it does not convert the input to the primitive cell, which would be rhombohedral in this case.

The output for the fluorite example is similar to the following. First the header with some information about the system, the version (the commit number), and the location of the relevant library and density files:

```
                  _   _     _          ___
                 (_) | |   (_)        |__ \
     ___   _ __  _  | |_   _    ___     ) |
    / __| | '__| | | | __| | |  / __|   / /
   | (__  | |    | | | |_  | | | (__   / /_
    \___| |_|    |_|  \__| |_|  \___| |____|

* CRITIC2: analysis of real-space scalar fields in solids
  (c) 1996-2013 A. Otero-de-la-Roza, A. Martin-Pendas,
                V. Luana
  Distributed under GNU GPL v.3 (see COPYING for details)
  Bugs, requests, and rants: alberto@fluor.quimica.uniovi.es
  If you find this software useful, please cite:
  Comput. Phys. Commun. 185 (2014) 1007-1018.
  Comput. Phys. Commun. 180 (2009) 157-166.

+ critic2, commit 8758a50
 compile host: Linux xxxx
 compile date: Thu May 21 19:54:25 MDT 2015
    using f77: ifort -g
          f90: ifort -g -FR -fopenmp -g -CU -C -traceback
               -fpe0 -debug -fpp -check all -check
               noarg_temp_created -O0
      ldflags:
       debug?: yes
 compiled dat: /usr/local/share/critic2
      datadir: /home/alberto/git/critic2/dat
     dic file: /home/alberto/git/critic2/dat/cif_core.dic
...was found?:  T

CRITIC2--2015/5/25, 13:06:26.273
```

After the CRYSTAL keyword is read, critic2 first lists the basic information about the crystal, starting with the cell parameters and the number of atoms in the crystal motif:

```
* Input crystal data
  Lattice parameters (bohr): 10.323763  10.323763  10.323763
  Lattice parameters (ang): 5.463100  5.463100  5.463100
  Lattice angles (degrees): 90.000  90.000  90.000
  Molecular formula: Ca(1) F(2)
  Number of non-equivalent atoms in the unit cell: 2
  Number of atoms in the unit cell: 12
  Number of electrons in the unit cell: 152
```

Next comes the non-equivalent atom list. In this case, the whole crystal is generated by replicating two atoms: one Ca and one F. The positions, multiplicities, and the atomic numbers are indicated:

```
+ List of non-equivalent atoms:
     x          y          z      name mult  Z
  -------------------------------------------
   0.00000    0.00000    0.00000  Ca    4    20
   0.25000    0.25000    0.25000  F     8    9
  -------------------------------------------
```

The next table is the complete atom list. Here, critic2 lists all the atoms in the unit cell: 4 Ca and 8 F. The exact same list is repeated in Cartesian coordinates, referred to the internal coordinate framework used in critic2.

```
+ List of atoms in the unit cell (cryst):
       x               y               z           name
  -------------------------------------------------------
   0.0000000000   0.0000000000   0.0000000000      Ca
   0.0000000000   0.5000000000   0.5000000000      Ca
   0.5000000000   0.0000000000   0.5000000000      Ca
   0.5000000000   0.5000000000   0.0000000000      Ca
   0.2500000000   0.2500000000   0.2500000000      F
   0.2500000000   0.7500000000   0.7500000000      F
   0.7500000000   0.2500000000   0.7500000000      F
   0.7500000000   0.7500000000   0.2500000000      F
   0.7500000000   0.7500000000   0.7500000000      F
   0.7500000000   0.2500000000   0.2500000000      F
   0.2500000000   0.7500000000   0.2500000000      F
   0.2500000000   0.2500000000   0.7500000000      F
  -------------------------------------------------------

+ List of atoms in the unit cell (bohr):
       x               y               z           name
  -------------------------------------------------------
     0.00000000      0.00000000      0.00000000    Ca
     0.00000000      5.16188142      5.16188142    Ca
     5.16188142      0.00000000      5.16188142    Ca
     5.16188142      5.16188142      0.00000000    Ca
     2.58094071      2.58094071      2.58094071    F
     2.58094071      7.74282213      7.74282213    F
     7.74282213      2.58094071      7.74282213    F
     7.74282213      7.74282213      2.58094071    F
     7.74282213      7.74282213      7.74282213    F
     7.74282213      2.58094071      2.58094071    F
```

```
      2.58094071     7.74282213     2.58094071      F
      2.58094071     2.58094071     7.74282213      F
   -------------------------------------------------------
```

Following this information comes the cell volume, in atomic units and in cubed angstrom:

```
+ Cell volume (bohr^3): 1100.30746
+ Cell volume (ang^3): 163.04874
```

And then the list of symmetry operations:

```
+ List of symmetry operations (48):
  Operation 1:
     1.000000  0.000000  0.000000  0.000000
     0.000000  1.000000  0.000000  0.000000
     0.000000  0.000000  1.000000  0.000000
  Operation 2:
     0.000000  0.000000 -1.000000  0.000000
    -1.000000  0.000000  0.000000  0.000000
     0.000000 -1.000000  0.000000  0.000000
[...]
  Operation 48:
     0.000000  1.000000  0.000000  0.000000
     0.000000  0.000000  1.000000  0.000000
    -1.000000  0.000000  0.000000  0.000000

+ List of centering vectors (4):
  Vector 1: 0.000000  0.000000  0.000000
  Vector 2: 0.000000  0.500000  0.500000
  Vector 3: 0.500000  0.000000  0.500000
  Vector 4: 0.500000  0.500000  0.000000

+ Centering type (p=1,a=2,b=3,c=4,i=5,f=6,r=7): 6
```

The Cartesian/crystallographic transformation matrices are the transformation operations between the vector basis formed by the cell vectors (crystallographic coordiantes) and the internal Cartesian axes used in critic2 (Cartesian coordinates). The crystallographic to cartesian matrix ("crystocar") gives the cell vectors in Cartesian axes. The metric tensor is the crystocar matrix times its transpose.

```
+ Cartesian/crystallographic transformation matrices:
  Car to crys (xcrys = A * xcar)
       0.0968639067     0.0000000000     0.0000000000
       0.0000000000     0.0968639067     0.0000000000
       0.0000000000     0.0000000000     0.0968639067
```

```
Crys to car (xcar = B * xcrys)
    10.3237628366     0.0000000000     0.0000000000
     0.0000000000    10.3237628366     0.0000000000
     0.0000000000     0.0000000000    10.3237628366
Metric tensor
   106.5800791057     0.0000000000     0.0000000000
     0.0000000000   106.5800791057     0.0000000000
     0.0000000000     0.0000000000   106.5800791057
```

Some more information about crystal symmetry follows, including the list of operations in chemical notation (and their principal axes), the crystal point group, the Laue class and the cyrstal system:

```
+ Symmetry operations in chemical notation:
  1    E    (  0.00000 ,  0.00000 ,  0.00000 )
  2    S3   ( -0.57735 , -0.57735 , -0.57735 )
  3    C4   (  0.00000 ,  0.00000 ,  1.00000 )
  4    C2   (  0.70711 ,  0.70711 ,  0.00000 )
  5    C2   (  0.00000 ,  1.00000 ,  0.00000 )
  6    C2   (  0.00000 ,  0.00000 ,  1.00000 )
  7    C2   (  0.70711 , -0.70711 ,  0.00000 )
  8    C4   (  0.00000 ,  0.00000 ,  1.00000 )
  9    C2   (  1.00000 ,  0.00000 ,  0.00000 )
  10   S4   (  0.00000 ,  1.00000 ,  0.00000 )
  11   s    (  0.70711 ,  0.00000 ,  0.70711 )
  12   S3   ( -0.57735 ,  0.57735 ,  0.57735 )
  13   S3   ( -0.57735 ,  0.57735 , -0.57735 )
  14   s    ( -0.70711 ,  0.00000 ,  0.70711 )
  15   S4   (  0.00000 ,  1.00000 ,  0.00000 )
  16   S3   ( -0.57735 , -0.57735 ,  0.57735 )
  17   C3   ( -0.57735 , -0.57735 , -0.57735 )
  18   C4   (  1.00000 ,  0.00000 ,  0.00000 )
  19   C4   (  1.00000 ,  0.00000 ,  0.00000 )
  20   C3   (  0.57735 ,  0.57735 , -0.57735 )
  21   C3   ( -0.57735 ,  0.57735 ,  0.57735 )
  22   C2   (  0.00000 , -0.70711 ,  0.70711 )
  23   C2   (  0.00000 ,  0.70711 ,  0.70711 )
  24   C3   (  0.57735 , -0.57735 ,  0.57735 )
  25   i    (  0.00000 ,  0.00000 ,  0.00000 )
  26   s    (  0.70711 ,  0.70711 ,  0.00000 )
  27   S4   (  0.00000 ,  0.00000 ,  1.00000 )
  28   s    (  0.00000 ,  1.00000 ,  0.00000 )
  29   s    (  0.00000 ,  0.00000 ,  1.00000 )
  30   s    ( -0.70711 ,  0.70711 ,  0.00000 )
```

```
     31   S4   (  0.00000 ,  0.00000 ,  1.00000 )
     32   s    ( -1.00000 ,  0.00000 ,  0.00000 )
     33   C3   (  0.57735 ,  0.57735 ,  0.57735 )
     34   C4   (  0.00000 ,  1.00000 ,  0.00000 )
     35   C2   (  0.70711 ,  0.00000 ,  0.70711 )
     36   C3   ( -0.57735 ,  0.57735 ,  0.57735 )
     37   C3   (  0.57735 , -0.57735 ,  0.57735 )
     38   C2   (  0.70711 ,  0.00000 , -0.70711 )
     39   C4   (  0.00000 ,  1.00000 ,  0.00000 )
     40   C3   ( -0.57735 , -0.57735 ,  0.57735 )
     41   S3   ( -0.57735 , -0.57735 , -0.57735 )
     42   S4   (  1.00000 ,  0.00000 ,  0.00000 )
     43   S4   (  1.00000 ,  0.00000 ,  0.00000 )
     44   S3   ( -0.57735 , -0.57735 ,  0.57735 )
     45   S3   ( -0.57735 ,  0.57735 ,  0.57735 )
     46   s    (  0.00000 ,  0.70711 , -0.70711 )
     47   s    (  0.00000 ,  0.70711 ,  0.70711 )
     48   S3   ( -0.57735 ,  0.57735 , -0.57735 )
+ Crystal point group: Oh
+ Number of operations (point group): 48
+ Laue class: m-3m
+ Crystal system: cubic
```

The next few lines give the number of atoms contributing to the density in the main cell (the cell at the lattice origin). This is the set of atoms around the main cell whose in vacuo atomic density contribution to the main cell is more than a certain threshold. These "atomic environments" are used in certain applications of critic2 that involve quantites that break the translational symmetry of the crystal (e.g. calculating the promolecular density at a point in space).

```
+ Atoms contributing density to the main cell:
  Number of atoms: 15972
```

A list of nearest-neighbor shells for all atoms in the non-equivalent list follows, together with information about the nearest-neighbor distance, the faces of the Wigner-Seitz cell (these are used to calculate distances between non-equivalent atoms), and, finally, whether the input cell is orthogonal:

```
+ Atomic environments:
   id   atom   nneig     distance     nneq  type
  ------------------------------------------------
   1     Ca      8       4.4703204   2    F
          ...   12       7.3000027   1    Ca
          ...   24       8.5600119   2    F
          ...    6      10.3237628   1    Ca
          ...    24      11.2500597   2    F
          ...    24      12.6439756   1    Ca
          ...    32      13.4109613   2    F
          ...    12      14.6000054   1    Ca
          ...    48      15.2690512   2    F
          ...    24      16.3233023   1    Ca
    2      F      4       4.4703204   1    Ca
          ...     6       5.1618814   2    F
          ...    12       7.3000027   2    F
          ...    12       8.5600119   1    Ca
          ...     8       8.9406409   2    F
          ...     6      10.3237628   2    F
          ...    12      11.2500597   1    Ca
          ...    24      11.5423177   2    F
          ...    24      12.6439756   2    F
          ...    16      13.4109613   1    Ca
  ------------------------------------------------

+ List of nearest neighbor distances:
   id   atom   rnn/2 (bohr)
  ---------------------------
   1     Ca      2.2351602
   2      F      2.2351602
  ---------------------------

+ Wigner-Seitz neighbors
   1:  0  0 -1
   2:  0 -1  0
   3:  1  0  0
   4: -1  0  0
   5:  0  1  0
   6:  0  0  1
+ Is the cell orthogonal? T
```

Critic2 always has a "reference" scalar field defined. The reference field is the one that, for instance, provides the attraction basins tha are integrated, or whose critical points are determined by AUTO. In absence of any external field loaded by the user, critic2 defaults to the promolecular density (the sum of atomic densities) calculated using the internal density tables. The information that follows in the output shows how critic2 builds the promolecular density. First, the atomic numbers and charges are identified, the number of electrons is counted, and then the density tables for the appropriate atoms are loaded from external files:

```
* Atomic radial density grids
```

```
+ List of atoms and charges:
   id  atom    Z    Q  ZPSP
  ---------------------------
    1   Ca    20    0   -1
    2    F     9    0   -1
  ---------------------------

+ Number of electrons: 152
+ Number of valence electrons: 152
+ Number of core electrons: 0

+ Reading new promolecular density grids:
+ Read density file: /home/alberto/git/critic2/dat/ca_pbe.wfc
  Log grid (r = a*e^(b*x)) with a = 1.2394E-04, b = 2.0000E-03
  Num. grid points = 5855, rmax (bohr) = 15.0633965
  Integrated charge = 19.9999732987
  El. conf.: 1S(2)2S(2)2P(6)3S(2)3P(6)4S(2)
+ Read density file: /home/alberto/git/critic2/dat/f__pbe.wfc
  Log grid (r = a*e^(b*x)) with a = 2.7542E-04, b = 2.0000E-03
  Num. grid points = 5161, rmax (bohr) = 8.3542920
  Integrated charge = 8.9999953160
  El. conf.: 1S(2)2S(2)2P(5)
+ Reading new core density grids:
```

Finally, the just-built promolecular density is made available to the user through identifier number 0, and it is set as reference. A list of the current integrable properties is also given. This is the list of properties that would be integrated in the attraction basins if the user runs INTEGRALS or any of the other integration methods:

```
* Field number 0 is now REFERENCE.

* Integrable properties list
   Id  Type  Field  Name
  -------------------------
    1   v        0  Volume
    2  fval      0  Charge
    3  lval      0  Lap
  -------------------------
```

The execution finishes with a report of the warnings found and the timestamp. It is always a good idea to check for warnings in the output.

```
CRITIC2 ended succesfully (0 WARNINGS, 0 COMMENTS)

CRITIC2--2015/5/25, 13:06:32.168
```

In most of the critic2 keywords, atoms can be selected by their atomic symbol ("at.s" in the syntax definitions), in which case the keyword applies to all atoms with the same atomic number unless otherwise stated. Atoms can also be selected by an integer identifier from the non-equivalent atom list ("nat.i" in the definitions) in those cases in which symmetry makes it irrelevant which of the symmetry-equivalent atoms in the cell are used. In the example above, nat.i = 1 is Ca and nat.i = 2 is F. For example, the non-equivalent atom identifier can be used to instruct critic2 to calculate the atomic population in the Ca basin. Because all Ca are equivalent by symmetry, it is not necessary to specify which Ca atom we want.

In the cases in which symmetry can not be used (for instance, if we want to represent the basin of an atom at a particular location), then critic2 reads an integer identifier from the complete atom list ("at.i" in the following). In the example above, at.i = 1 represents the Ca at (0 0 0), at.i = 2 is the Ca at (0.5 0.5 0), and so on.

In exact parallel to the atomic lists, critic2 also maintains a list of non-equivalent critical points (CP) and a complete (or cell) list of critical points found for the current reference scalar field. The CPs in the non-equivalent list reproduce all the CPs in the complete list by symmetry. In the keyword definitions the same rules for atoms apply to the CP identifiers: "ncp.i" for non-equivalent CPs and "cp.i" for complete-list CPs.

Since the atoms are always critical points, the non-equivalent (complete) atom list is a subset of the non-equivalent (complete) CP list. Critic2 makes sure that the integer identifier for all atoms in atomic lists is the same as the identifier for the same atom in the CP list. Hence, in the example above, cp.i = 2 can be used to select the critical point associated to Ca (0.5 0.5 0) in the complete CP list because at.i = 2 is that atom in the complete atom list.

Some additional notation particularities that will be used in the rest of the manual:

- By **scalar field** or **field** we mean a numerical or analytical representation of a function that associates a scalar value to every point in space. Most of the time, this function is the electron density, for which special techniques are provided (for instance, core augmentation in the case of valence densities, see ZPSP below). However, critic2 can deal with any arbitrary field: the ELF, the Laplacian,...

- The **promolecular density** is the scalar field built by using the sum of the in-vacuo atomic densities. This object comes up in a number of contexts. For instance, NCIPLOT and HIRSHFELD, or in the calculation of the deformation density: the real electron density minus the promolecular density. The promolecular density does not require any input from the user other than the crystal structure, and is always available using field identifier 0.

- We denote by <root> the root of the input file. That is, the name of the file without the extension. If no input file is known (for instance, because critic2 was run as 'critic2 < inputfile'), then the root defaults to "stdin". The default root can be changed with the keyword ROOT.

- The critical points of a field can be classified by their **rank** (r) and **signature** (s). The rank is the number of non-zero eigenvalues of the Hessian. In the vast majority of cases, r = 3. The signature is the number of positive eigenvalues minus the number of negative eigenvalues. s = -3 is a maximum, s = -1 is a first-order saddle point, s = +1 is a second-order saddle point and s = 3 is a minimum. All these critical points necessarily appear in a periodic field and receive special names: nuclear CP, bond CP, ring CP, and cage CP, respectively. The abbreviations ncp, bcp, rcp, and ccp are also used throughout the manual. Note that a maximum is a 'nuclear critical point' even though it may not be associated to any nucleus.

4 Arithmetic expressions, variables, and functions in critic2

4.1 Basic usage

In critic2, an arithmetic expression can be used almost everywhere in the input where a real or integer number is expected. Arithmetic expressions that appear in the input (without an associated keyword) are evaluated and their result is written to the output. For instance, you can start critic2 and write:

```
3+2*sin(pi/4)
(i) 3+2*sin(pi/4)
(o)    4.4142135623731
```

In the same way, variables can be defined and utilized in any expression. Variable names must start with a letter and be composed only of letters, numbers, and the underscore character. Also, they can not have the same name as a known constant (pi, e, eps) or a function, regardless of case. Variables in expressions are case-sensitive. To use a variable, first you need to assign it. For instance:

```
a = 20+10
a/7 + 1
(i) a/7 + 1
(o)    5.2857142857143
```

By using the "-q" command-line option, critic2 can be used as a simple calculator:

```
$ echo "1+erf($RANDOM/100000)" | critic2 -q
1.2578249310340
```

When used in combination with other keywords, arithmetic expressions must be enclosed in either double quotes ("), single quotes ('), or parentheses, or they must form a single word (i.e. no spaces). For instance, this is valid critic2 input:

```
a = 0.12
NEQ 1/3 "2 / 3" 1/4+a Be
```

but this is not:

```
NEQ 1/3 2 /3 1/4 Be
```

Arithmetic expressions can contain:

- Operators: +, -, , /, * or ^, % (modulo)
- Functions: see List of available functions.
- Constants: pi, e and eps (the machine precision).
- Variables defined by the user as above.

Parentheses can be used and the usual rules of associativity and precedence apply.

In some cases, arithmetic expressions can be applied to transform a scalar field. Scalar fields are denoted with a dollar sign ($) followed by an optional identifier and the field number. The latter corresponds to the order in which the field was loaded in the input. For instance, if the first field ($1) is the spin-up density and the second ($2) is the spin-down density, the total density can be calculated with the expression "$1+$2". The expression "$0" represents the promolecular density (which is always available once the crystal structure is known) so "$1+$2-$0" would represent the density difference between the actual crystal density and the sum of atomic densities.

Between the $ and the field number, it is possible to specify an identifier to access the derivatives of the field. The identifier can be:

- v: valence-only value of the field (it is usually employed to access the valence density in a grid field in which core augmentation is active).
- c: core-only value of the field.
- x, y, z: first derivatives.
- xx, xy, xz, yy, yz, zz: second derivatives.
- g: norm of the gradient.
- l: Laplacian.
- lv: valence Laplacian (Laplacian without core augmentation).
- lc: core Laplacian.

Therefore, for instance "$l2" is the Laplacian of field 2 and "$xy3" is the xy-component of the Hessian of field 3.

There is a clear divide between expressions that reference fields and those that do not and, for certain keywords, critic2 will decide what to do with an expression based on this distinction. For instance:

```
a = 2
CUBE CELL FIELD "a*$1"
```

calculates a grid spanning the entire cell for a scalar field that is built as two times the field number 1, but:

```
a = 2
CUBE CELL FIELD "a*1"
```

uses field number two (a*1 = 2) to calculate the same grid. Expressions involving fields can also be used in LOAD and in other keywords (POINT, LINE, INTEGRABLE, etc.).

The value of a variable can be cleared using the CLEAR keyword:

```
CLEAR var1.s var2.s ...
CLEAR ALL
```

This keyword deletes the variables var1.s, var2.s, etc. or all the variables (ALL). At any moment, the internal list of variables can be printed to the output using the keyword LIST:

```
LIST
```

4.2 List of available functions

Arithmetic expressions can use any of the functions in the critic2 function library. These functions include the usual mathematical functions (like exp or sin) but also functions that are meant to be applied to scalar fields of a certain type (e.g., the Thomas-Fermi kinetic energy density, gtf). It is very important to distinguish whether a function expects a numerical argument (e.g. sin(x)), an integer field identifier (e.g. gtf(id)).

The list of arithmetic functions is: abs, exp, sqrt, floor, ceil, ceiling, round, log, log10, sin, asin, cos, acos, tan, atan, atan2, sinh, cosh, erf, erfc, min, max. All these functions apply to numbers (or other arithmetic expressions), and their behavior is the usual. For instance, "sin(2*pi)", "max($1,0)", and "atan2(y,x)" are all valid expressions.

The following "chemical" functions accept one or more integer field identifiers as their arguments. Their purpose is to provide shorthands to build fields from other fields using physically-relevant formulas. For instance, "gtf(1)" is the Thomas-Fermi kinetic energy density for field 1, which is assumed to be the electron density for the system. In all instances, "gtf(1)" is equivalent to writing the formula in full ("3/10*(3*pi^2)^(2d0/3d0)*$1^(5/3)") but, obviously, much more convenient. The library of chemical functions includes:

- gtf(id) [GTF]: Thomas-Fermi kinetic energy density. The kinetic energy density for a uniform electron gas with its density given by the value of field id at the point. The argument field can be of any type. See:
  - Yang and Parr, Density-Functional Theory of Atoms and Molecules.
- vtf(id) [VTF]: the potential energy density calculated using the Thomas-Fermi kinetic energy density and the local virial theorem (2g(r) + v(r) = 1/4*lap(r) in au).
- htf(id) [HTF]: the total energy density calculated using the Thomas-Fermi kinetic energy density and the local virial theorem (2g(r) + v(r) = 1/4*lap(r) in au).
- gtf_kir(id) [GTF_KIR]: Thomas-Fermi kinetic energy density with the semiclassical gradient correction proposed by Kirzhnits for the not-so-homogeneous electron gas. The electron density and its derivatives are those of field id at every point in space. See:
  - Kirzhnits, (1957). Sov. Phys. JETP, 5, 64-72.
  - Kirzhnits, Field Theoretical Methods in Many-body Systems (Pergamon, New York, 1967).
  - Abramov, Y. A. Acta Cryst. A (1997) 264-272.

  and also:

  - Zhurova and Tsirelson, Acta Cryst. B (2002) 58, 567-575.
  - Espinosa et al., Chem. Phys. Lett. 285 (1998) 170-173.

  for more references and an example of the application of this quantity with experimental electron densities.
- vtf_kir(id) [VTF_KIR]: the potential energy density calculated using gtf_kir(id) and the local virial theorem (2g(r) + v(r) = 1/4*lap(r) in au).
- htf_kir(id) [HTF_KIR]: the total energy density calculated using gtf_kir(id) and the local virial theorem (2g(r) + v(r) = 1/4*lap(r) in au).
- gkin(id) [GKIN]: the kinetic energy density, G-version (grad(rho) * grad(rho)). The argument should be a wfn/wfx field. See:
  - Bader and Beddall, J. Chem. Phys. (1972) 56, 3320.
  - Bader and Essen, J. Chem. Phys. (1984) 80, 1943.
- kkin(id) [KKIN]: the kinetic energy density, K-version (rho * lap(rho)). The argument should be a wfn/wfx field.

- lag(id) [LAG]: the Lagrangian density (-1/4 * lap(rho)).
- vir(id) [VIR]: the electronic potential energy density, also called the virial field.
  - Keith et al. Int. J. Quantum Chem. (1996) 57, 183-198.
- he(id) [HE]: the electronic energy density, vir(id) + gkin(id).
- elf(id) [ELF]: the electron localization function (ELF). The argument should be a wfn/wfx field. See:
  - Becke and Edgecombe J. Chem. Phys. (1990) 92, 5397-5403

The name in square brackets is a shorthand for applying the chemical function to the reference field (case-insensitive) in the POINTPROP keyword.

A particular case of chemical function is xc(), that allows the user to access the external LIBXC library. This is only possible if the library was linked during the compilation of critic2. See Use of LIBXC in arithmetic expressions.

4.3 Use of LIBXC in arithmetic expressions

If critic2 is linked to the libxc library (see the README for instructions on how to do this) then the xc() function can be used in arithmetic expressions. xc() calculates the exchange and/or correlation energy density for one of the functionals in the libxc library. The number of arguments to xc() depends on the type of functional invoked, which is selected using an integer index. The list of functionals available and their corresponding indices should be consulted in the libxc documentation. The integer index that selects the functional always appears *last* in the calling sequence of xc(). For ease of reference, the file libxc_funcs_2.0.1.txt, specifying the list of functionals in version 2.0.1 of libxc, is included in the doc/ directory.

The arguments to xc(...,idx) depend on the type of functional specified by the idx integer, which can be:

- LDA: xc(rho,idx)
- GGA: xc(rho,grad,idx)
- meta-GGA: xc(rho,grad,lapl,tau,idx)

where rho is the electron density expression, grad is its gradient, lapl is its Laplacian and tau is the kinetic energy density. Note that rho, grad, lalp, and tau are *expressions*, not integer indices like in the chemical functions above. For instance, the expression for LDA using the electron density loaded in field number 1 would be:

```
xc($1,1)+xc($1,9)
```

because idx=1 is Slater's exchange and idx=9 is Perdew-Zunger correlation. PBE (a GGA) would be:

```
xc($1,$2,101)+xc($1,$2,130)
```

Here 101 is PBE exchange and 130 is PBE correlation. Field $1 contains the electron density and $2 is its gradient, which can be determined using, for instance:

```
LOAD AS GRAD 1
```

5 The crystal structure

5.1 Loading a crystal structure (CRYSTAL/MOLECULE)

```
CRYSTAL file.cif [datablock.s]
CRYSTAL file.cube
CRYSTAL file.struct
CRYSTAL [file.]POSCAR [at1.s at2.s ...|POTCAR]
CRYSTAL [file.]CONTCAR [at1.s at2.s ...|POTCAR]
CRYSTAL [file.]CHGCAR [at1.s at2.s ...|POTCAR]
CRYSTAL {[file.]CHG|[file.]ELFCAR} [at1.s at2.s ...|POTCAR]
CRYSTAL file_DEN
CRYSTAL file.OUT # e.g. elk's GEOMETRY.OUT
CRYSTAL file.out # e.g. ESPRESSO's file.scf.out (PW output)
CRYSTAL file.in  # e.g. ESPRESSO's file.scf.in (PW input)
CRYSTAL file.STRUCT_IN
CRYSTAL file.STRUCT_OUT
MOLECULE file.xyz [border.r] [CUBIC|CUBE]
MOLECULE file.wfn [border.r] [CUBIC|CUBE]
MOLECULE file.wfx [border.r] [CUBIC|CUBE]
CRYSTAL
  SPG spg.s
  CELL a.r b.r c.r alpha.r beta.r gamma.r
  CARTESIAN [scal.r]
    # comment
    [BOHR/AU]
    [ANGSTROM/ANG]
    x1.r y1.r z1.r
    x2.r y2.r z2.r
    x3.r y3.r z3.r
  ENDCARTESIAN/END
  NEQ x.r y.r z.r at.s [ZPSP zpsp.i] [Q q.i]
```

```
       [ANG/ANGSTROM] [BOHR/AU]
    CENTER|CENTRE [x.r y.r z.r]
  ENDCRYSTAL/ENDMOLECULE/END
  CRYSTAL LIBRARY label.s
  # Note: all CRYSTAL keywords can be replaced by MOLECULE
  # and viceversa, with the effect discussed below.
```

The first part of every critic2 input usually specifies the crystal structure. Critic2 can read a large number of crystal structue formats. In the simplest usage, critic2 reads one of the popular crystal structure file formats like, for instance, a cif file. This is done by using:

```
  CRYSTAL file.s
```

The extension of file.s is used to determine the appropriate reading format. At present, critic2 understands cif files (extension .cif), Gaussian cube files (.cube), WIEN2k's struct files (.struct), abinit density files (_DEN), elk's GEOMETRY.OUT (the .OUT extension is used, regardless of whether the file name is GEOMETRY.OUT or not), Quantum ESPRESSO's pw.x output (.out), Quantum ESPRESSO's pw.x input (.in), VASP's POSCAR, CONTCAR, CHGCAR, ELFCAR, and CHG, siesta's STRUCT_IN (input) and STRUCT_OUT (generated by the calculation), molecular xyz files, and Gaussian's wfn and wfx files. Some relevant format-specific comments:

- Gaussian cube files are assumed to represent a periodic system, even though they come from a molecular calculation. This is not a limitation provided the emcompassing cell is large enough and, in actuality, critic2 is prepared to handle the particular case of a gas-phase molecule by using the MOLECULE keyword instead of CRYSTAL. See section Molecular structures.

  Despite their name, cube files may represent non-orthogonal cells, have become quite a standard, and can be written by a lot of mainstream solid-state programs, so it is a very good option if critic2 provides no native interface to the densities of your program of choice.

- WIEN2k's struct files may or may not contain the symmetry operations (for instance, if they have been generated by the web interface but not run). If symmetry is provided, then critic2 will use it. Otherwise, critic2 will try to guess the symmetry (see Symmetry options).

- Depending on the version, VASP's POSCAR and CONTCAR (and also CHG and CHGCAR) may be missing the atomic symbols. In that case, they need to be provided by hand after the file name, either using a list of atomic symbols or the location of the POTCAR. Even though it is the name accepted by VASP, files that have the extension POSCAR, CONTCAR, CHG and CHGCAR are also accepted (e.g. nacl.POSCAR). Note that the pseudopotential charges are NOT read from the POTCAR.

- Any file with extension .OUT will be understood to be in elk's GEOMETRY.OUT format, even though it may not be called GEOMETRY.

- Regarding Quantum ESPRESSO outputs (usually file.scf.out, only the .out extension is detected), the block at the beginning of the run is read (from 'Title:' ... to 'Cartesian axes'). Because this block is repeated at the end of a geometry optimization ('a final calculation at the relaxed geometry', it says), the final structure from a successfully completed geometry optimization will be read. However, if the optimization crashed, then the first structure is read.

- Quantum ESPRESSO (PW) inputs can also be read (usually file.scf.in but only the .in extension is detected).

- A gas-phase molecule can be input using an xyz file or a Gaussian wfn/wfx file. These file formats can be read using the CRYSTAL keyword, but MOLECULE will make critic2 select the appropriate default options for molecules in a supercell. The input molecule is enclosed in a parallelepiped that is larger (default: 3 bohr) in all directions than the minimal parallelepiped encompassing the molecule. If the CUBIC (or CUBE) keyword is given, then a cubic supercell is used. The width of the vacumm around the molecule can be changed with the optional border.r argument (in bohr). The molecule is automatically translated to the center of the cube. See the section below (Molecular structures) for more details.

- CIF files are read using the ciftbx library by Sydney R. Hall. If the CIF file provides the symmetry operations, they are read. The specific data block to read from a multi-block cif file can be specified with the optional argument datablock.s. For instance, to read the data_shelx block, use datablock.s = "shelx". If datablock.s is not present, then the first block is read.

If none of the file formats above is available, the crystal structure can be specified "by hand". This is done using the CRYSTAL environment. For instance, the input for magnesium oxide (rocksalt structure) would be:

```
CRYSTAL
  SPG F m -3 m
  CELL 4.213 4.213 4.213 90 90 90 ANG
  NEQ 0. 0. 0. Mg
  NEQ 1/2 1/2 1/2 O
ENDCRYSTAL
```

There are several relevant keywords in the CRYSTAL environment. A particular space group can be used with the SPG keyword, in which case only the atoms in the asymmetric unit need to be given:

```
SPG spg.s
```

The SPG keyword builds the complete set of symmetry operations in the crystal from the space group label alone. The space group names known to critic2 correspond to those given in the International Tables for Crystallography vol. A, in the Hermann-Mauguin notation. Each element in the label is separated by one or more blanks, and the input is case insensitive. Examples of values for spg.s are "F m -3 m", "P b a 2", "P 1 n 1", "P 63 m c" (without the quotes).

If two origin choices are possible, then an additional token "1" or "2" can be appended to the space group label to choose between them. If no origin choice is specified, origin 1 is used by default (note that in most electronic structure programs, origin 2 is default, however). For example: "F d d d 1" (F d d d with origin choice 1), "F d d d 2" (same group with origin choice 2), "F d d d" (same as origin 1).

For R space groups, hexagonal and rhombohedral axes are possible. These can be specified by an "h" or "r" token after the space group label. If no "h" or "r" is given, hexagonal axes are used. For the hexagonal, the obverse cell is used: (0 0 0), (1/3,2/3,2/3), and (2/3,1/3,1/3). For instance, "R 3 h" (hexagonal axes), "R 3 r" (rhombohedral axes), "R 3" (same as "R 3 h").

For monoclinic groups, when several unique axis choices (where all have the same label), the tokens "b" and "c" choose between the unique axis b and c respectively. By default, the b unique axis is used. Example: "P 2 b" (unique axis b, same as "P 1 2 1"), "P 2 c" (unique axis c, same as "P 1 1 2"), "P 2" (same as "P 2 b" and "P 1 2 1").

The space group label can also be given as a single word without spaces. For instance, "Fm-3m" is the same as "F m -3 m". The space group number can be used as well, in which case all defaults for origin choice, unique axis and hexagonal/rhombohedral axes apply. Example: "222" is equivalent to "Pn-3n", "Pn-3n1", and "P n -3 n 1".

If no SPG keyword is found in the CRYSTAL environment, then an internal routine calculates the symmetry from the atomic positions (which can be deactivated using NOGUESS, see Symmetry options). In this case, all the atoms in the cell need to be given (in contrast to using SPG, in which case only the asymmetric unit needs to be given). There are two possible ways to input the cell parameters. In the simplest approach, the CELL keyword can be used to give the cell lengths and angles:

```
CELL a.r b.r c.r alpha.r beta.r gamma.r [ANG/ANGSTROM/BOHR/AU]
```

If the ANG (or ANGSTROM) keyword is used, then a.r, b.r and c.r are in angstrom. Otherwise, cell parameters are in bohr by default (the keyword BOHR or AU can be used, but it is redundant). Using the CELL keyword lets critic2 decide on the crystallographic-to-Cartesian transformation matrix (which is not unique; critic2 always uses the Cholesky decomposition of the metric tensor, for consistency). Alternatively, one could input this matrix using the CARTESIAN keyword:

```
CARTESIAN [scal.r]
```

```
    # comment
    [BOHR/AU]
    [ANGSTROM/ANG]
    x1.r y1.r z1.r
    x2.r y2.r z2.r
    x3.r y3.r z3.r
ENDCARTESIAN/END
```

CARTESIAN reads the Cartesian coordinates of the cell vectors in some arbitrary orthonormal reference frame . Each row corresponds to a vector (hence the metric tensor is G = R * R' where R is the matrix above and ' is the transpose, and the coordinate transformation is (cryst) = (cart) * R). If scal.r is given, all vectors are scaled by that factor. The input units can be controlled with the BOHR/AU and ANG/ANGSTROM keywords (default: bohr).

The CARTESIAN keyword is particularly useful for inputs based on Quantum ESPRESSO results since the matrix corresponds exactly to the CELL_PARAMETERS matrix, and scal.r can be set to celldm(1).

The NEQ keyword can be used to specify the atomic positions:

```
NEQ x.r y.r z.r at.s [ZPSP zpsp.i] [Q q.i] [ANG/ANGSTROM]
    [BOHR/AU]
```

NEQ adds one atom to the crystal. If symmetry was given (via SPG), then only the non-equivalent atom list needs to be given (as in the MgO example above), which decreases the number of NEQs that are necessary. Otherwise, the complete list of atoms in the cell has to be given, and the symmetry guess will reduce it to the non-equivalent atoms list. The coordinates x.r y.r z.r are crystallographic (fractional) coordinates unless the ANG/ANGSTROM or BOHR/AU keywords are used. Usage of Cartesian coordinates requires a previous CELL or CARTESIAN in order to convert the atomic positions to crystallographic. The atomic symbol is at.s, and the atomic number is detected from the symbol (essentially, from the beginning of it, so atoms like Cl1 or baxx2 are valid). Deuterium (D) is detected as hydrogen.

In addition to the atomic position and identity, the atomic charge (Q) and pseudopotential charge (ZPSP) can be given. The former is used to calculate the point-charge electrostatic energy (using Ewald's method, see Commands for crystallographic computation) and, if Q is positive, to build the promolecular density using the corresponding cation instead of the neutral atom.

The role of ZPSP is to augment the fields defined on a grid, a procedure called core augmentation. ZPSP is the pseudopotential atomic number, that is, the atomic number of the atom minus the number of electrons represented by the pseudopotential. For instance, ZPSP=2 for Ba. The normal use of critic2 with grids involves loading the valence density or pseudo-density that sometimes needs to be augmented to recover an electron density that has maxima at the nuclear positions, which is important for certain applications (for instance, plotting). While this procedure is not satisfactory from the theoretical point of view, in practice, the valence regions are mostly unaffected. Of course, core augmentation needs to be deactivated if the field is not the electron density or if the grid already contains the core contributions, and it is usually a better option to use the full all-electron density from the electronic structure code if it is able to reconstruct it.

The value of ZPSP can be found easily in a VASP calculation by grepping 'ZVAL' in the POTCAR file. In Quantum ESPRESSO, this information is in the UPF file (the number of valence electrons). The ZPSP and Q values are not set when the structure is read from a file except in the case of abinit files. The standalone versions of ZPSP and Q can be used outside the CRYSTAL environment (see below).

The MOLECULE keyword can be used as an alternative to CRYSTAL to deal with molecular grids and wavefunctions. Also, cube files calculated with Gaussian (e.g. with cubegen) can be used in critic2. The program will assume periodic boundary conditions but topologies, graphical representations and computations with cubes can be performed nevertheless (see the "Molecular structures" section below).

When using a molecule as input (for instance, from an xyz file using the ANGSTROM keyword), the CENTER keyword can be useful:

```
{CENTER|CENTRE} x.r y.r z.r
```

CENTER displaces the crystallographic coordinates of all atoms by x.r, y.r and z.r. The typical scenario for this keyword is the calculation of promolecular densities from an xyz file: in order to displace the molecule to the center of the periodic cell by using 'CENTER 0.5 0.5 0.5'. Note that it is not a good idea to use this keyword with, for instance, a cube file input, since the field and the geometry are going to be out of sync. Also, note that molecules read from xyz, wfn, and wfx files are automatically centered (by 1/2 1/2 1/2).

5.2 The crystal library

A library of (simple) crystal structures is provided with critic2, and can be accessed using the CRYSTAL LIBRARY keyword like this:

```
CRYSTAL LIBRARY NaCl
```

Critic2 tries to find the "NaCl" label (case-insensitive) in an internal library file that is distributed with the code, and usually located in dat/crystal.lib relative to the source of the distribution. The relevant library entry in this case reads:

```
STRUCTURE B1 rock_salt rocksalt NaCl
  CRYSTAL
    SPG f m -3 m
    CELL 5.6402 5.6402 5.6402 90 90 90 ANG
    NEQ 0.0 0.0 0.0 na
    NEQ 0.5 0.5 0.5 cl
  ENDCRYSTAL
```

```
ENDSTRUCTURE
```

Note the four aliases (B1, rock_salt, rocksalt, and NaCl) that are defined for the same crystal structure. The syntax in the CRYSTAL environment is exactly the same as above. The user can define their own library of crystal structures either by modifying the crystal.lib directly or by using the LIBRARY keyword:

```
LIBRARY /path/to/library
```

where the indicated file is used instead to perform the lookup. The path can be absolute or relative to the execution directory.

5.3 Symmetry options

Critic2 has two symmetry modules: "spg" and "guess". Spg accepts a label (for instance, "P m -3 m") and builds the symmetry from it. guess reads the unit cell description (lengths, angles, and the complete list of atoms in the cell) and calculates all the symmetry operations. Depending on the type of CRYSTAL input, the symmetry behavior defaults to:

- cif: uses the symmetry operations inside the file ("_symmetry_equiv_pos_as_xyz" or "_space_group_symop_operation_xyz" labels).
- cube: guess.
- struct: internal symmetry operations if present. Otherwise, guess.
- vasp: guess.
- abinit: internal information in the DEN file.
- elk: guess.
- espresso: guess.
- cell/cartesian/neq: spg if used. Otherwise, guess.

The treatment of symmetry in critic2 can be controlled using the GUESS/NOGUESS (before CRYSTAL) and CLEARSYM (after CRYSTAL) keywords.

```
GUESS
NOGUESS
```

Activates (GUESS) or deactivates (NOGUESS) the use of guess. They must be used before crystal to be effective. The default is GUESS.

```
CLEARSYM
```

Clear all symmetry operations (that is, use space group P 1). It must be used after CRYSTAL to be effective.

5.4 Atomic charge options

The Q (or QAT) and ZPSP keywords can be used to change the atomic charge and pseudopotential charge after CRYSTAL:

```
{Q|QAT|ZPSP} {nat1.i|at1.s} q1.r {nat2.i|at2.s} q2.r ...
```

Changes the atomic charge (Q, QAT) or the pseudopotential atomic number (ZPSP) of the non-equivalent atom i1.i, i2.i,... or the atom type at1.s, at2.s,.... Real numbers for the charges are acceptable but they are internally converted to integers. See the discussion above (Loading a crystal structure (CRYSTAL/MOLECULE)) for an explanation on the behavior of Q and ZPSP.

```
NOCORE
```

Clears the ZPSP values of all atoms.

5.5 Loading multiple crystal structures

Critic2 can only work with one crystal structure at a given time. However, more than one crystal can be loaded in the same run, simply by giving a new CRYSTAL (or MOLECULE) keyword. When the second and subsequent CRYSTAL keywords are read, critic2 clears all the information for the previous crystal structure, including the structural data (cell parameters and atomic positions) as well as all the fields that had been loaded, all the critical point information, and the atomic and core density grids. Effectively, critic2 behaves as if starting a new run, except in that the variables and global options are carried over. To clear the variables and set the global options to their default values, use the RESET keyword:

```
RESET
```

5.6 Molecular structures

Critic2 can be used to analyze scalar fields in isolated molecules as well as in crystals. This behavior is activated by using the MOLECULE keyword to load the structure, instead of the usual CRYSTAL. The MOLECULE keyword is most often used when loading an xyz or a Gaussian wfn/wfx file.

Because critic2 works under periodic boundary conditions, it does the analysis of molecular structures in the xyz/wfn/wfx format by placing translating the input molecule to the center of a supercell large enough to encompass it. The structures loaded from a format different from xyz/wfn/wfx are not translated. The amount of border around the molecule can be controlled (see border.r in "The crystal structure" section above) as well as the shape of the supercell (CUBIC/CUBE). Provided the size of the vacuum is large enough, the analysis can be carried out without problems.

The use of MOLECULE instead of CRYSTAL changes some of the default behavior in critic2. Namely:

- The symmetry module is automatically deactivated. All molecular structures are run in the P1 space group.
- The default CP search seeding strategy is modified (see Finding critical points). In a crystal, a recursive subdivision of a symmetry-reduced portion of the Wigner-Seitz cell is used (SEED WS with DEPTH 1). In a molecule, the default is to seed at the center of every interatomic line between atom pairs less than 15 bohr apart (SEED PAIR).
- In addition to the supercell, a second smaller cell is defined, the "molecular cell", that can be represented using the MOLCELL keyword in WRITE (Exporting the crystal structure (WRITE)) or CPREPORT (Finding critical points). The molecular cell is only used in molecules loaded from a xyz/wfn/wfx file. The region outside the molecular cell is assumed to belong to the vacuum. Any CPs found outside the molecular cell are discarded, and all downwards gradient paths that exit the molecular cell are assumed to have diverged to infinity.

In the (unusual) case in which you want to analyze a molecular structure not coming from a xyz/wfn/wfx, critic2 will not automatically translate or set up a molecular cell. In this case, you can use the MOLCELL keyword to set it up:

```
MOLCELL [border.r]
```

The MOLCELL keyword calculates the smallest parallelepiped encompassing the molecular motif and then adds a border to it in order to build the molecular cell. The default border is 3 bohr, but can be controlled by passing an additional numerical argument. Note that using this keyword only makes sense if the molecule is placed close to the center of the cell and if there is a sizable amount of vacuum between the molecule and the cell edges. In order to use MOLCELL, the input structure needs to be read using the MOLECULE keyword and the cell needs to be orthogonal.

5.7 Exporting the crystal structure (WRITE)

Critic2 can be used as a converter between different crystal structure file formats. For instance, if we are given a cif file that needs to be converted to a QE input, we use:

```
CRYSTAL myfile.cif
WRITE myfile.scf.in
```

Sometimes, it is also useful to create a finite representation of the crystal by taking the crystal motif, perhaps extended with some of atoms in the neighboring cell. The simplest way of doing this is by writing an xyz file:

```
CRYSTAL myfile_DEN
WRITE myfile.xyz MOLMOTIF
```

The MOLMOTIF keyword writes all atoms in the unit cell and completes the molecules by using the atoms in the neighboring cells. A covalent radii criterion is used to determine whether two atoms are bonded or not. The full syntax of the WRITE keyword is:

```
WRITE file.{xyz,gjf} [SPHERE rad.r [x0.r y0.r z0.r]]
      [CUBE side.r [x0.r y0.r z0.r]]
      [BORDER] [MOLMOTIF] [BURST] [ix.i iy.i iz.i]
WRITE file.{obj,ply,off} [SPHERE rad.r [x0.r y0.r z0.r]]
      [CUBE side.r [x0.r y0.r z0.r]] [BORDER] [MOLMOTIF] [BURST]
      [CELL] [MOLCELL] [ix.i iy.i iz.i]
WRITE file.scf.in [PRIMITIVE]
WRITE file.tess
WRITE file.incritic
WRITE {[file.]POSCAR|[file.]CONTCAR} [PRIMITIVE]
WRITE file.abin [PRIMITIVE]
WRITE file.elk [PRIMITIVE]
WRITE file.cif
WRITE file.m
WRITE file.gin [DREIDING]
WRITE file.lammps
WRITE file.fdf
WRITE file.STRUCT_IN
```

At present, a number of formats can be written by critic2, although it is easy to add more to this list and it is likely to increase in the future. As in CRYSTAL, the type of file is detected by the extension (xyz, in, tess, incritic, etc.).

In the first WRITE command above, an xyz file containing a finite piece of the crystal is generated. Alternatively, if the gjf extension is used, a template for a Gaussian input file is written. The number of cells used in each direction is given by ix.i, iy.i and iz.i (default: 1, 1, 1). For the purpose of its graphical representation, it is sometimes interesting to include atoms which are almost exactly at the edge of the cell. For instance, when doing:

```
CRYSTAL
  SPG f m -3 m
  CELL 5.64 5.64 5.64 90 90 90 ANG
  NEQ 0. 0. 0. na
  NEQ 1/2 1/2 1/2 cl
ENDCRYSTAL
```

```
WRITE nacl.xyz
```

Critic2 will (correctly) generate a list of 4 Na and 4 Cl atoms, representing 1/8th of the conventional cell, because the atoms at (1, 0, 0), (1, 1/2, 0), etc. are repetitions of the atoms in the main cell. However, this does not look good when the unit cell is represented graphically, because many of the atoms in the cubic cell are "missing". By using BORDER, these atoms are included.

The keyword MOLMOTIF is used in molecular crystals. All atoms in the requested cells are written to the xyz file. Then, the molecules that lie across the cell boundary are completed using atoms from the neighboring cells. In order to do this, a connectivity criterion is used: two atoms are bonded if their distance is less than 1.5 times their sum of covalent radii.

The BURST keyword is similar to MOLMOTIF in that the molecular crystal structure is partitioned into individual molecules, which are then completed using the atoms from neighboring cells. However, in contrast to MOLMOTIF, each of those molecular fragments is written to a separate file.

The SPHERE keyword writes all atoms inside a sphere of radius rad.r (in bohr) and centered around the crystallographic coordinates (x0.r y0.r z0.r). If no center is given, (0,0,0) is used. The similar keyword CUBE writes all atoms inside a cube of side side.r centered around (x0.r y0.r z0.r) (default: (0,0,0)).

There is an important application of the xyz-format WRITE keyword: the coordinates written to the xyz file are consistent with the transformation to Cartesian coordinates in critic2, so it is possible to give back all or part of these coordinates to critic2 in order to represent a subset of the atoms in the crystal. This is incredibly useful when generating fragments for an NCIPLOT calculation (see FRAGMENT keyword in section Non-covalent interaction visualization (NCIPLOT)).

The OBJ output is similar to xyz: critic2 will write a finite piece of the crystal to a file with Wavefront OBJ format. The OBJ format is a three-dimensional model representation, that is, it uses vertices and faces instead of atoms. This file format is understood by many visualizers such as view3dscene, meshlab, blender, and others. The keywords have the same meaning as in the xyz format. The additional CELL keyword instructs critic2 to write a stick representation of the unit cell. In the case of a molecular structure (loaded using the MOLECULE instead of the CRYSTAL keyword), the MOLCELL keyword can be used to represent the "molecular cell", that is, the region around the molecule outside which critic2 considers critical points and gradient paths to have diverged to infinity. The similarly popular PLY (polygon file format or Stanford triangle format) and OFF (Geomview) file formats can be used as well, with the same options.

Quantum ESPRESSO inputs can be written using the extension scf.in. This conversion is especially useful in the case of low-symmetry crystals (e.g. monoclinic in a non-conventional setting) where the conversion from other formats, such as CIF, can be tricky. The QE input generation works by first determining the Bravais lattice from the symmetry operations. Critic2 uses 'ibrav=0' always, and writes a CELL_PARAMETERS block containing the crystallographic-to-Cartesian transformation matrix. QE is particular about how this matrix should written in order for its own symmetry module to work. If the crystal setting matches any of those covered in the QE manual, then that particular matrix is used. Otherwise, critic2 uses its own internal CELL_PARAMETERS matrix, which may result in Quantum ESPRESSO failing to recognize the crystal symmetry.

A tessel input (extension .tess) and a critic2 input using the CELL/NEQ mechanism (.incritic) can be written.

A VASP POSCAR (or CONTCAR) can be generated by using the POSCAR/CONTCAR extension or name. The list of atomic types is written to the critic2 output. This list is necessary to build the corresponding POTCAR. The atoms are always ordered in increasing atomic number.

An abinit input file containing the input structure can be written by using the 'abin' extension. An elk input template can be written using the elk extension. A simple cif file (no symmetry) is generated if the cif extension is used.

The octave script file (extension .m) contains the crystal structure in octave format and is prepared to be read using escher (visit gatsby.ucmerced.edu for more information).

A simple GULP template input file containing the structure (and EEM as the first line) can be written using the .gin extension. If the DREIDING keyword is used, critic2 will attempt to find the atom types in the DREIDING force field. No resonant atoms are detected. For file names with an extension .lammps, critic2 writes a simple LAMMPS data file containing one unit cell (length units are angstrom). Only orthogonal cells are supported for now. Both the GULP and the LAMMPS outputs are experimental, so please exercise care and double-check the templates.

Two types of siesta inputs can be generated. The fdf extension writes a template for a proper functional siesta input template containing the crystal structure. The STRUCT_IN extension writes a files that can be read using the MD.UseStructFile option.

To make critic2 try to reduce the current cell to the primitive cell before writing the template file, use the PRIMITIVE keyword. The code that does this is still not very well polished, particularly for the more "exotic" centering types, so I recommend you check your resulting structure with a visualization program. In the case of QE, critic2 tries to conform as much as possible with the QE rules for writing the CELL_PARAMETERS matrix.

6 Scalar fields

6.1 Loading a field (LOAD)

Scalar field, functions that associate a value to every point in real space, are loaded with the LOAD keyword. A number of different field formats are supported:

```
    LOAD file.cube
```

```
LOAD {file.DEN|file_DEN}
LOAD [file.]CHGCAR
LOAD {[file.]CHG|[file.]ELFCAR}
LOAD file.qub
LOAD file.xsf
LOAD file.grid
LOAD file.clmsum file.struct
LOAD file.{RHO,BADER,DRHO,LDOS,VT,VH}
LOAD file.OUT GEOMETRY.OUT
LOAD file.OUT GEOMETRY.OUT OTHER.OUT
LOAD file.wfn
LOAD file.wfx
LOAD file1.ion {nat1.i/at1.s} file2.ion ...
LOAD COPY id.i [TO id2.i]
LOAD PROMOLECULAR
LOAD PROMOLECULAR file.xyz
LOAD [WIEN|ELK|PI|CUBE|ABINIT|VASP|VASPCHG|QUB|XSF|ELKGRID] file
LOAD ... [NEAREST/TRILINEAR/TRISPLINE/EXACT/APPROXIMATE/
          RHONORM/VNORM/CORE/NOCORE/NUMERICAL/ANALYTICAL
          TYPNUC {-3,-1,1,3}/NORMALIZE n.r]
LOAD AS "expression.s" [n1.i n2.i n3.i|SIZEOF id.i|GHOST]
LOAD AS PROMOLECULAR {n1.i n2.i n3.i|SIZEOF id.i} [file.xyz]
LOAD AS CORE {n1.i n2.i n3.i|SIZEOF id.i}
LOAD AS LAP id.i
LOAD AS GRAD id.i
LOAD AS CLM {ADD id1.i id2.i|SUB id1.i id2.i}
```

Critic2 loads scalar fields in "field slots": integer identifiers that represent the field throughout the run. There are 31 slots, numbered from 0 up to 30. At the beginning of the run, slot number 0 is occupied by the promolecular density, which is automatically loaded right after the crystal structure is succesfully read. Successive LOAD commands fill the slots in increasing order: the first LOAD will assign the new field to slot 1, the second LOAD will occupy slot 2 and so on. By default, the first loaded field (other than the promolecular density in slot 0) becomes the reference field (see The reference field).

The simplest usage of load is:

```
LOAD file.ext
```

The field information is read from file.ext. Critic2 uses the extension to decide which format should be used for the reading:

- cube for Gaussian cube files (grid).
- DEN for abinit grids.

- CHGCAR, CHG, or ELFCAR for vasp grids.
- qub for aimpac grids.
- xsf for xcrysden grids.
- grid for elk's grids (this requires a patch to elk, e-mail to request it).
- RHO, BADER, DRHO, LDOS, VT, and VH for siesta's grids.
- clmsum for WIEN2k.
- OUT for elk's STATE.OUT.
- ion for aiPI ion files.
- wfn and wfx for Gaussian wavefunction files.

In the case of WIEN2k and elk, not all the necessary information is encapsulated in the clmsum or STATE.OUT (for instance, the muffin tin radii are missing), so it is necessary to provide a second file: the struct file or the GEOMETRY.OUT respectively. This is the case, regardless of whether these same files were used in the CRYSTAL keyword.

Gaussian wavefunction files can be loaded by passing the wfn or wfx file to critic2. This option is normally used in combination with MOLECULE on the same file in order to obtain the same structure from the isolated molcule. The evaluation of wfn/wfx densities is analytical (no grids involved).

For aiPI inputs, several ion files are necessary. The ion files can be associated to atoms either by using their non-equivalent atom number (nat1.i) or their atomic symbol (at1.s). The keyword PROMOLECULAR is used to load a promolecular density field (same as the one in slot 0). If a finite fragment of the crystal is passed to LOAD PROMOLECULAR as an xyz file (file.xyz), then the sum of atomic densities is built using only the atoms in that fragment.

Elk's STATE.OUT is version-dependent. The following versions of elk have been tested and work with critic2: 2.3.22, 2.3.16, 2.2.10, 2.2.9, 2.2.8, 2.2.5, 2.2.1, 2.1.25, 2.1.22, 1.4.22, 1.4.18, 1.4.5, 1.3.31, 1.3.30, 1.3.24, 1.3.22, 1.3.20, 1.3.15, 1.3.2, 1.2.20, 1.2.15, 1.1.4, 1.0.17, 1.0.16, and 1.0.0. If your version is not supported (most likely because it's newer; I tend to revise the part of the code that reads elk files every aeon or so), please contact me.

VASP fields can come in two varieties. The CHGCAR and AECCAR* files give the grid values in a higher precision and multiplied by the cell volume. The CHGCAR can be read directly with LOAD and the AECCAR can be read with something like:

```
LOAD VASP AECCAR0
```

The other format is the CHG, that is also used for the ELFCAR (containing the values of the ELF function). In this case, the grid values are given with less precision and they may or may not be multiplied by the cell volume (depending on the VASP version). Both can be loaded with:

```
LOAD CHG
LOAD ELFCAR
```

But if you have a file in this format that does not conform to those names, you can use the VASPCHG keyword:

```
LOAD VASPCHG STRANGE_CHG_FILE_NAME.chg
```

The CHGCAR files are relatively safe to use, but CHGs data points may or may not be multiplied by the unit cell volume, and you have to tell critic2 which one it is. If you have a CHG file that is multiplied by the volume, then you can force critic2 to load it in the same way as a CHGCAR with:

```
LOAD VASP CHG
```

I have never seen an ELFCAR that is multiplied by the volume (it would not make much sense anyway), so the default for ELFCAR files is to assume this.

Using a patched version of elk it is possible to generate other files (OTHER.OUT) containing the spherical harmonics/plane waves representation of other fields, such as the elf and the Coulomb potential. See the tools/elk_mod directory for the patch source files. When the OTHER.OUT (e.g. ELF.OUT) file is used, then that field is loaded instead of the density.[1]

In some cases, the extension of the file to be loaded may be different from the one that critic2 expects. For instance, WIEN2k creates files with extensions other than clmsum for different fields (e.g. clmup for the spin-up density) but the same format. In those cases, the keywords WIEN/ELK/... can be used to force critic2 to read in a specific format.

The LOAD COPY keyword can be used to make a copy of the field in slot id.i to the next available slot, or to slot id2.i if it is given with the keyword TO.

6.2 Additional options

The definition of a field can be supplemented by additional optional keywords that depend on the type of field in the input and that come after the file name. They are:

```
NEAREST/TRILINEAR/TRISPLINE
```

Choose the grid interpolation mode in a grid field (e.g. a cube file). NEAREST, use the field at the nearest grid point. TRILINEAR, trilinear interpolation. TRISPLINE, 3d-spline interpolation (adapted from the abinit code, this part was coded by A. Lherbier according to the source). If some derivatives are not available (first and second in NEAREST, second in TRISPLINE), they are taken as zero.

[1]N.B.: the code for grid and qub file extensions (all of them corresponding to grids) is not very well tested. Same thing goes for elk .OUT files with a third file different from the density (I think a dirty hack I did to elk once wrote these). If you are planning on using these fields, please send me an e-mail -- A.

By default, TRISPLINE is used but this may require a lot of memory in very large cube files (say, 400^3). However, the memory allocation for the TRISPLINE interpolation only happens if the parts of critic2 that require derivatives or interpolation at points outside of the grid are called. If only grid-based algorithms (e.g. YT, BADER, NCIPLOT with default NSTEP,...) then no additional memory is used.

Applies to: grids. Default: TRISPLINE.

CORE/NOCORE

The electron density from a pseudopotential/plane-waves calculation, given on a grid, only represents valence electrons. In order to get an approximation to the all-electron density, it can be augmented by summing the corresponding core contributions to the electron density at the atomic sites. The ZPSP values are used to determine how many electrons are added by each core. NOCORE deactivates this behavior (i.e., critic2 will use only the valence density from the external grid file).

Applies to: grids (it can be activated for any other field using CORE, but that would not make much sense). Default: CORE. Note that for most grid fields read with LOAD, the pseudopotential charges (ZPSP) are not set, so even if CORE is active by default, in practice, no core augmentation is done, because critic2 does not know how many core electrons to add. Hence, the effective behavior is *not* to core-augment, despite CORE being active. The exception to this are the grid fields loaded from an abinit DEN file, from which the pseudopotential charges (ZPSP) are automatically extracted and set.

EXACT/APPROXIMATE

The calculation of the electron density in aiPI fields is relatively expensive. Using the APPROXIMATE keyword, the atomic contributions to the density are calculated by interpolating from a radial grid, that is precomputed at the beginning of the run. EXACT calculates the fully analytical values of the field with no approximations.

Applies to: PI. Default: APPROXIMATE.

RHONORM/VNORM

In WIEN2k, the clmsum and other files representing the electron density have different normalization. The l=0 m=0 radial component is divided by sqrt(4*pi). The option VNORM can be used if the file in the input does not present this normalization (e.g. a potential file).

Applies to: WIEN2k. Default: RHONORM.

NUMERICAL

Calculate the derivatives of the field numerically. Mostly used for testing the implementation of new field types.

Applies to: all. Default: no.

ANALYTICAL

Calculate the derivatives of the field analytically. Opposite of NUMERICAL.
Applies to: all. Default: yes.

```
TYPNUC {-3,3}
```

Controls whether the nuclei are maxima (-3) or minima (3) of the field.
Applies to: all. Default: -3 (maxima).

```
NORMALIZE n.r
```

Normalize the grid integral over the unit cell to n.r.
Applies to: grids.

6.3 Field arithmetics

New fields can be defined as combinations of the existing ones using the LOAD AS keyword. This allows the creation of fields that are not directly computed by the electronic structure program, and is very useful in combination with the cube-writing facilities (keyword CUBE in Simple graphical representations: points, lines, planes, grids). For instance, if rhoup.cube and rhodn.cube are the spin-up and spin-down densities, it is possible to get the total density and the spin density using:

```
LOAD rhoup.cube
LOAD rhodn.cube
LOAD AS "$1+$2"
LOAD AS "$1-$2"
```

The spin-up density is loaded as field 1, rhodn.cube is field 2, field 3 is defined as the sum of both and field 4 is the spin density, which can, for instance, be graphically represented or integrated in the atomic basins to get the atomic magnetic moments (see INTEGRABLE in Integration of atomic basins).

The list of arithmetic operations that can be performed on fields is quite large (see Arithmetic expressions, variables, and functions in critic2). If no additional keywords are present between LOAD AS and the arithmetic expression (like LAP, GRAD, or CLM, see below), then two types of fields can be loaded. If the arithmetic expression involves at least one grid, the resulting field will be a grid with the same number of points. If more than one grid appears in the expression, then the new grid size is the maximum of the number of points in every dimension. If the expression contains no grid fields, then the resulting field is what critic2 calls a "ghost field". A ghost field is just an arithmetic expression that is parsed and processed every time the field is evaluated. The analyitcal derivatives for a ghost field are, naturally, not available, so the NUMERICAL option is the default for this kind of field (see Additional options).

Regardless of the types of fields in the expression passed to LOAD AS, a grid field can be enforced by explicitly specifying the size of the grid, either by giving the number of points in each direction (n1.i n2.i n3.i) or by adopting the size of another grid (SIZEOF, the mold grid has field id.i). If grid fields appear in the expression but you want a ghost field, then use the GHOST keyword.

The PROMOLECULAR option to LOAD AS allows the creation of a grid out of the promolecular density, with number of points n1.i, n2.i, n3.i or the size of grid field number id.i if the SIZEOF keyword is used. Similarly, CORE creates a grid using only the core densities, as specified by the ZPSPs of the atoms. If a fragment of the crystal is passed as an xyz file to any of those keywords, then only the atoms in the fragment contribute to the sum of atomic (or core) densities.

The LAP, GRAD and CLM keywords of LOAD AS apply only to specific types of fields:

- GRAD only applies to grid fields. It defines a new grid as the norm of the gradient of the grid field with index id.i.

- LAP applies to grid, WIEN2K, and elk fields. In all cases, the keyword creates a field of the same type containing the Laplacian of the field with index id.i. If id.i is a grid, the Laplacian is calculated by Fourier transform. If it is a WIEN2k or elk field, the Laplacian is calculated analytically both inside the muffin tins and in the interstitial using the appropriate transformation.

- CLM defines a new field as the addition (id1+id2) or substraction (id1-id2) of fields of the WIEN2K or elk type by using the ADD and SUB keywords respectively. Note that the muffin tin radii, number of plane-waves, etc. have to be the same for both source fields.

6.4 Changing the field options after LOAD

The options of a given field can be changed anywhere in the input after it has been loaded using the SETFIELD keyword:

```
SETFIELD id.i [NEAREST/TRILINEAR/TRISPLINE/
               EXACT/APPROXIMATE/
               RHONORM/VNORM/
               CORE/NOCORE/
               NUMERICAL/ANALYTICAL/
               TYPNUC {-3,-1,1,3}/
               LAP|NOLAP]
```

SETFIELD changes the properties of the field in slot id.i. The keywords have the same meaning as in the previous sections.

Where: after the corresponding LOAD.

6.5 Unloading a field

```
UNLOAD {id.i|ALL}
```

Unload the field in slot id.i or all fields (keyword ALL).
Where: after LOAD.

6.6 The reference field

One of the loaded fields is chosen as the **reference** field. The reference field is used as the primary field in all computations (representing its basins with BASINPLOT, integrations, plotting, etc.) unless otherwise specified. The first field loaded becomes the reference field. If no fields have been loaded, then the reference is the promolecular density (field 0). In order to change the reference field, the REFERENCE keyword can be used:

```
REFERENCE id.i
```

Sets field number id.i as reference.

7 Usual CRYSTAL/LOAD combinations

- **WIEN2k**

```
CRYSTAL name.struct
LOAD name.clmsum name.struct
```

- **Quantum ESPRESSO**

```
CRYSTAL name.cube
LOAD name.cube
ZPSP C 4 H 1 ...

CRYSTAL name.scf.out
LOAD name.cube
ZPSP C 4 H 1 ...

CRYSTAL
  CARTESIAN celldm(1)
    ... CELL_PARAMETERS matrix ...
  ENDCARTESIAN
  NEQ x0 y0 z0 C ZPSP 4
  ... rest of the ATOMIC_POSITIONS list ...
```

```
ENDCRYSTAL
LOAD name.cube

CRYSTAL name.scf.in
LOAD name.cube
ZPSP C 4 H 1 ...
```

- **Abinit**

```
CRYSTAL name_DEN
LOAD name_DEN
```

- **VASP CHGCAR format (divided by cell volume)**

```
CRYSTAL POSCAR POTCAR
LOAD CHGCAR ZPSP C 4 H 4

CRYSTAL POSCAR POTCAR
LOAD AECCAR0
LOAD AECCAR2
LOAD AS "$1+$2"
REFERENCE 3

CRYSTAL POSCAR POTCAR
LOAD CHG
LOAD ELFCAR
```

- **elk**

```
CRYSTAL GEOMETRY.OUT
LOAD STATE.OUT GEOMETRY.OUT
```

- **aiPI**

```
CRYSTAL
  CELL a b c alpha beta gamma
  SPG 'space group'
  NEQ x y z atom
  ...
ENDCRYSTAL
LOAD at1.ion at1 at2.ion at2 ...
```

- **Any code that writes Gaussian cubes**

```
CRYSTAL name.cube
LOAD name.cube
```

- **Gaussian molecules**

```
MOLECULE name.wfn
LOAD name.wfn

MOLECULE name.wfx
LOAD name.wfx
```

- **Promolecular/procrystal density calculations**

```
CRYSTAL name.cif

MOLECULE name.xyz
```

- **Conversion between crystal formats**

```
CRYSTAL name.cif
WRITE name.scf.in
WRITE name.xyz MOLMOTIF
```

- **Conversion between grid formats**

```
CRYSTAL name.cif
LOAD rhoup_DEN
LOAD rhodn_DEN
LOAD AS '$1-$2' n1 n2 n3
CUBE GRID FILE rhospin.cube FIELD 3
```

8 Simple graphical representations: points, lines, planes, grids

8.1 Points (POINT)

```
POINT x.r y.r z.r [ALL] [FIELD {id.i|"expr.s"}]
```

Calculates the value of the reference field, its derivatives, and related quantities at the point (x.r, y.r, z.r) in crystallographic coordinates. If ALL is used, all loaded fields are evaluated. In addition, all arithmetic expressions that have been registered using the POINTPROP keyword are also calculated (see List of properties calculated at points (POINTPROP)). The POINTPROP keyword combined with POINT is useful to evaluate chemical functions at arbitrary points in space.

If FIELD is used and followed by an integer (id.i), then only that field is evaluated. FIELD followed by an arithmetic expression calculates the value of that expression at the point.

8.2 Lines (LINE)

```
LINE x0.r y0.r z0.r x1.r y1.r z1.r npts.i [FILE file.s]
     [FIELD id.i|"expr.s"] [GX|GY|GZ|GMOD|HXX|HXY|HXZ|HYX|HYY|
     HYZ|HZX|HZY|HZZ|LAP]
```

Calculate a line from (x0.r y0.r z0.r) to (x1.r y1.r z1.r) with npts.i points. By default, the result is written to the standard output, but it can be redirected to a file using FILE. The reference field is used unless a FIELD keyword appears, in which case the field id.i or the expression expr.s are evaluated. Together with the value of the field, an additional quantity is evaluated: the components of the gradient (GX,GY,GZ), the norm of the gradient (GMOD), the components of the Hessian (HXX,...) and the Laplacian (default: Laplacian) of the reference (or the id.i) field.

8.3 Planes and contour plots (PLANE)

```
PLANE x0.r y0.r z0.r x1.r y1.r z1.r nx.i ny.i ...
PLANE {ATOM|ATOMS} at1.i at2.i at3.i [{SCALE sx.r sy.r}|
      {SIZE zx.r zy.r}] nx.i ny.i ...

PLANE ... [FILE file.s] [FIELD id.i|"expr.s"]
     [F|GX|GY|GZ|GMOD|HXX|HXY|HXZ|HYX|HYY|HYZ|HZX|HZY|HZZ|LAP]
     [CONTOUR {LOG niso.i|ATAN niso.i|BADER|LIN niso.i r0.r r1.r}]
```

Calculate the value or other properties of the reference field on a plane. The results are written to a file, with default name <root>_plane.dat. There are two ways to specify the geometry of the plane. In the first option, three points are given: (x0.r y0.r z0.r) is the origin, (x1.r y1.r z1.r) is the x-end of the plane and (x2.r y2.r z2.r) is the y-end. The number of calculated points on each axis are given by nx.i (x-axis) and ny.i (y-axis).

An alternative way of passing the plane geometry to critic2 is to use the ATOM (or ATOMS) keyword. This keyword accepts three integers, corresponding to the atom identifiers in the atomic cell list. The sx.r and sy.r parameters to SCALE enlarge the plot plane by sx.r in the x direction and by sy.r in the y direction. The total length of the x or y axis can be set with SIZE (in bohr). SCALE and SIZE are mutually exclusive. The syntax of ATOM is similar to GRDVEC. The rest of the options FILE, FIELD, ... are treated in the same way.

Regardless of how the plane geometry is given, there are a number of additional options that control the behavior of PLANE. The name of the output file can be changed with FILE. Using FIELD, one of the loaded fields (id.i) or an expression ("expr.s") can be evaluated. In addition to the field value, a second property can be evaluated: the field again (F), its derivatives (Gx), its second derivatives (Hxx), the gradient norm (GMOD) or the Laplacian (LAP).

The keyword CONTOUR writes a contour map representation of the plane: two contour line files (.iso and .neg.iso) and a gnuplot script (.gnu). The isovalue distribution can be: logarithmic (LOG, with niso.i contours), arctangent (ATAN, with niso.i contours), the original aimpac distribution (BADER, {1,2,4,8}x10^{-3,-2,-1,0,1}) and linear (LIN, niso.i contours from r0.r to r1.r). The field property selected with [F|GX|...] is used for the contour plot.

The GRDVEC keyword (see Gradient path representations in a plane (GRDVEC)) performs the same functions as PLANE with the CONTOUR option, and more (like, for instance, tracing gradient paths in the plane), but it is a little more complex to use.

8.4 Grids (CUBE)

```
CUBE x0.r y0.r z0.r x1.r y1.r z1.r {nx.i ny.i nz.i|bpp.r} ...
CUBE CELL {nx.i ny.i nz.i|bpp.r} ...
CUBE GRID ...

CUBE ... [FILE file.s] [FIELD id.i/"expr.s"]
     [F,GX,GY,GZ,GMOD,HXX,HXY,HXZ,HYY,HYZ,HZZ,LAP]
```

The CUBE keyword writes a three-dimensional grid in Gaussian cube or VASP CHGCAR formats. The limits of the grid can be set in three ways. By giving the end-points (x0.r y0.r z0.r) and (x1.r y1.r z1.r) in crystallographic coordinates it is possible to build a grid from an orthogonal fragment of the crystal. The CELL keyword calculates a grid spanning the entire unit cell, which may or may not be orthogonal depending on the crystal structure. GRID has the same effect as CELL regarding the output grid geometry.

The number of points in the grid can also be controlled in several ways. If the cube limits are given explicitly or using CELL, then the number of points in each axis can be indicated by giving three integers (nx.i, ny.i, and nz.i) corresponding to the number of points in the x-, y-, and z-axis respectively. If a single number (bpp.r) is found, then the number of points is the length of the axis divided by bpp.r (that is, bpp is the number of bohrs per point, hence the name). The GRID keyword can be used to write a field defined on a grid directly to a cube file. This is useful when combined with the LOAD keyword to read, manipulate, and then save grids to an external file. If GRID is used, both the geometry of the grid and the number of points are adopted from the corresponding field.

Independently on how the grid is set up, several options control the behavior of CUBE. FILE sets the name of the output file (default: <root>.cube). If the extension of file.s is not .cube, then critic2 uses the vasp-style CHGCAR format for the output. FIELD sets the field slot number to be used. Alternatively, an arithmetic expression that combines the existing fields can be used. Finally, a derivative of the scalar field (gradient, Hessian, Laplacian) can be selected instead of the value of the field itself (F) to build the grid.

9 Finding critical points

9.1 Automatic determination of critical points (AUTO)

```
AUTO [GRADEPS eps.r] [CPRHO rho.r] [CPEPS eps.r]
     [FILE file.s] [VERBOSE] [DRY] ...
AUTO ... [CLIP CUBE x0.r y0.r z0.r x1.r y1.r z1.r]
AUTO ... [CLIP SPHERE x0.r y0.r z0.r rad.r]
AUTO ... [SEED ...] [SEED ...] ...
AUTO SEED WS [DEPTH depth.i] [X0 x0.r y0.r z0.r]
             [RADIUS rad.r]
AUTO SEED OH [DEPTH depth.i]  [X0 x0.r y0.r z0.r]
             [RADIUS rad.r] [NR nr.r]
AUTO SEED SPHERE [X0 x0.r y0.r z0.r] [RADIUS rad.r]
                 [NTHETA ntheta.i] [NPHI nphi.i] [NR nr.r]
AUTO SEED PAIR [DIST dist.r] [NPTS n.i]
AUTO SEED TRIPLET [DIST dist.r]
AUTO SEED LINE [X0 x0.r y0.r z0.r] [X1 x0.r y0.r z0.r]
               [NPTS n.i]
AUTO SEED POINT  [X0 x0.r y0.r z0.r]
```

The search for the critical points (CP) of a scalar field (the points where the gradient of the field vanishes) is a basic task in QTAIM. In critic2, this search is almost always conducted using the automatic CP localization algorithm via the AUTO keyword.

The automatic search for critical points has two steps: seeding and searching. In the seeding step, a collection of points are selected in the unit cell that span the crystal or molecular space where CPs are likely to appear. In the search step, a Newton-Raphson algorithm is launched at each of the seeds in order to find nearby critical points.

The default seeding behavior in critic2 depends on whether the geometry under study is a crystal (loaded with the CRYSTAL keyword) or a molecule (MOLECULE keyword):

- In a crystal, AUTO calculates the Wigner-Seitz (WS) cell and its irreducible part (the smallest piece that reproduces the WS cell by symmetry). Once the irreducible WS

(IWS) cell is found, seed points are chosen by subdividing the edges, faces and interior of the tetrahedra that the IWS comprises up to a certain subdivision level (the DEPTH).

- In a molecule, a single seed is planted at the midpoint between every atom pair less than 15 bohr apart.

In addition, AUTO provides multiple seeding strategies that can be combined by the user using the SEED keyword. These include searches between pairs of atoms (PAIR), atomic triplets (TRIPLET), uniform seeding in a sphere (SPHERE), a recursive subdivision of an octahedron (OH), and seeding along lines (LINE) and at points (POINT). The seed list built using these "seeding actions" is pruned to remove duplicates. Optionally, a portion of the unit cell can be selected to restrict the search in real space using the CLIP keyword. Once the seed list is built, Newton-Raphson is applied at each of the seeds on the list, making full use of shared-memory parallelization and crystal symmetry.

The default seeding can be changed using one or more SEED keywords. When a SEED keyword is used, the default seeding strategy is forgotten by critic2 and manual control of the seeding is used instead. Several SEED keywords can be used at the same time, each one specifying a single "seeding action" that is determined by the keyword immediately after SEED. This keyword can be:

- WS: a recursive subdivision of the Wigner-Seitz cell to level depth.i (keyword: DEPTH, default: 1). The WS cell can be displaced and centered somewhere in the unit cell using X0 (cryst. coord., default: origin) and scaled down to a radius of rad.r bohr (keyword: RADIUS, default: not used).

- OH: a sphere of radius rad.r (keyword RADIUS, mandatory) is built around (x0.r y0.r z0.r) (keyword X0, cryst. coord., mandatory). On the surface of that sphere, points are set according to a recursive subdivision algorithm that starts from a single octahedron and uses depth.r recursion levels (keyword: DEPTH, default: 1). This is the same procedure as the TRIANG keyword in BASINPLOT. Each resulting point on the sphere surface determines a single ray along which nr.r seeds are uniformly distributed (keyword: NR, mandatory), from X0 to the surface of the sphere.

- SPHERE: a sphere of radius rad.r (keyword RADIUS, mandatory) is built around (x0.r y0.r z0.r) (keyword X0, cryst. coord., mandatory). On the surface of that sphere, points are uniformly distributed, with a placement algorithm that uses nphi.i points in the azimuthal angle (keyword: NPHI, mandatory) and ntheta.i points in the polar angle (keyword: NTHETA, mandatory). Each resulting point on the sphere surface determines a single ray along which nr.r seeds are uniformly distributed (keyword: NR, mandatory), from X0 to the surface of the sphere.

- PAIR: seeds are placed on the interatomic lines, for all atom pairs at a distance less than dist.r bohr (keyword: DIST, default: 15). The number of seeds per line is n.i (keyword: NPTS, default: 1).

- TRIPLET: seeds are placed at the barycenter of every atomic triplet in which the three atoms are at a distance from each other less than dist.r (keyword: DIST, default: 15).
- LINE: place n.i seeds (keyword: NPTS, default: 1) along a line between (x0.r y0.r z0.r) (keyword: X0, cryst. coords., default: origin) and (x1.r y1.r z1.r) (keyword: X1, cryst. coords., mandatory).
- POINT: place a single seed at (x0.r y0.r z0.r) (keyword: X0, cryst. coords., mandatory).

Multiple SEED keywords can be given in the same AUTO command. For instance:

```
AUTO SEED PAIR SEED WS SEED POINT 1/4 1/4 1/4
```

executes three seeding actions: a search between all atoms pairs (1 seed per pair), a recursive subdivision of the WS cell (one level), and a single seed at (0.25 0.25 0.25). The seed placement can be visualized using the optional VERBOSE keyword that writes an OBJ file (<root>_seeds.obj) containing the unit cell and all the seed positions.

The AUTO search can be restricted to a portion of the unit cell using the CLIP keyword. The CLIP keyword specifies a periodic region of real space. Only the seeds inside that region are used, and only the CPs found inside that region are accepted (although symmetry can replicate the CPs and send them outside the CLIP region; use CLEARSYM or NOGUESS to deactivate symmetry if necessary, see Symmetry options). There are two possible region shapes in CLIP: a parallelepiped (CUBE) and a sphere (SPHERE). The parallelepiped is specified by giving its initial (x0) and final (x1) points in crystallographic coordiantes. The sphere requires a center (x0 in cryst. coords.) and a radius (in bohr).

A number of additional optional keywords control the behavior of AUTO. GRADEPS is the gradient norm threshold for the optimization: if a CP is found with gradient norm less than GRADEPS (default: 1e-12), then it is accepted as CP.

If DRY (dry run) is used, then the seeding is done but the actual CP search is skipped. This is useful to examine the seed placement (in combination with VERBOSE) and also to print the current list of CPs at zero computational cost (for intsance, after a CHECK run).

CPRHO and CPEPS control the miniminum field value that a CP needs to have to be accepted and the minimum distance between CPs to consider them equivalent, respectively.

Because finding the CPs can be an expensive task in large low-symmetry crystals, the CP list can be saved to a file using the FILE keyword. The file.cps file is read in subsequent runs by using the same FILE syntax instead of calculating the CPs again.

Critic2 writes all the critical points found by AUTO to two internal lists: the "non-equivalent" list, containing only those CPs not equivalent by symmetry, and the "complete" list, which contains all the CPs in the unit cell (see Input format, output format, and notation). The most important part of the AUTO output is the "final report", which gives the non-equivalent CP list and some other useful information. Its appearance is (the table has been simplified to fit the width of the page):

```
* Critical point list, final report (non-equivalent cps)
  Topological class (n|b|r|c):   2(8) 1(16) 1(16) 2( 8)
  Morse sum :   0
# n pg type        position   mult name  f     |grad|     lap
1 Td  (3,-3) n 0.00 0.00 0.00  4 B  7.19E+1 0.00E+00 -3.00E+15
2 Td  (3,-3) n 0.25 0.25 0.25  4 P  2.36E+3 0.00E+00 -3.00E+15
3 C3v (3,-1) b 0.09 0.59 0.59 16 b1 1.25E-1 4.43E-17 -2.25E-01
4 C3v (3, 1) r 0.88 0.88 0.61 16 r1 1.08E-2 1.09E-16  3.45E-02
5 Td  (3, 3) c 0.75 0.75 0.75  4 c1 5.63E-3 1.34E-16  2.22E-02
6 Td  (3, 3) c 1.00 0.00 0.50  4 c2 7.28E-3 1.32E-16  2.58E-02
```

The non-equivalent CP list gives the type and position of all the non-equivalent CPs found, their associated name, rank and signature, and multiplicity. Critic2 also provides the site-symmetry (pg), and the values of the reference field (f), its gradient (grad) and its Laplacian (lap) evaluated at the CPs. The 'topological class' gives the number of non-equivalent ncp, bcp, rcp, and ccp found. The values in parentheses correspond to the total number of CPs of each class in the cell. If the list is complete, then the Morse sum is zero (in a crystal) or one (in a molecule). Sometimes, the zero (one) Morse sum condition is not fulfilled, which is usually caused by (often unavoidable) numerical shortcomings of the scalar field:

- In WIEN2k and elk densities, there might be spurious CPs at the surface of the muffin tin (where the density is discontinuous) that show up in the final report list. Every time a FPLAPW field is loaded, every atomic muffin is checked for discontinuities and the report printed to the output.
- In fields defined on a grid, the regions of small field (i.e. the crystal voids) are noisy, which creates a lot of spurious critical points. It is usual that a zero Morse sum is unobtainable in this case, as multitude of CPs tend to appear clustered around the position of the real CPs. The CPs where the value of the field is larger should be reliable, however. Using a fine grid, and relatively high CPEPS and a CPRHO usually helps to eliminate the uninteresting CPs.
- In scalar fields with extreme variations in value (e.g. the Laplacian of the electron density), it is unlikely that AUTO will find all the core CPs. A previous version of critic2 does that (available upon request) but since the core CPs are not all that interesting, we have decided to remove that code from this version.

The **complete CP list** is the list of all CPs in the unit cell. The identifiers from this list are used as input for other keywords (for instance, GRDVEC or FLUXPRINT) as they specify a particular position in the crystal. The entries are similar to the non-equivalent CP list (the table has been simplified):

```
* Complete CP list
```

```
    (x symbols are the non-equivalent representative atoms)
    n neq typ | x   | y   | z   | op.    (lvec+cvec)
  x 1  1  n    1.00 0.50 0.32  1   1.0   0.0   0.0
    2  1  n    0.50 0.00 0.67  2   0.0   0.0   1.0
  x 3  2  n    1.00 0.50 0.59  1   1.0   0.0   0.0
    4  2  n    0.50 0.00 0.40  2   0.0   0.0   1.0
  [...]
```

The columns are, in order, the CP identifier (n), the identifier for the same CP in the non-equivalent CP list (neq), the type of CP, the position in crystallographic coordinates, and the symmetry operation that transforms the CP from the non-equivalent CP list into the listed CP. 'Op.' corresponds to one of the symmetry operations listed in the output and "lvec+cvec" is a translation. Application of the operation 'op.' to the non-equivalent CP followed by the translation recovers the position of the CP in the complete list. The CPs that are at exactly the same position as the corresponding CPs in the non-equivalent list are listed first and marked with an 'x'. Note that for these, the operation is always 1 (the identity) and the translation vector is always a lattice translation.

9.2 Requesting more information about the critical point list

```
CPREPORT {SHORT|LONG|VERYLONG|SHELLS [n.i]|ESCHER}
CPREPORT {OBJ|PLY|OFF} [MOLMOTIF] [CELL] [MOLCELL]
         [BORDER] [ix.i iy.i iz.i]
```

CPREPORT prints additional information about the critical points to the output. SHORT: print the list of non-equivalent critical points with crystallographic coordinates, multiplicity, and properties of the reference scalar field. LONG: the complete list of critical points in the unit cell and the connectivity in the case of bcp and rcp (when the graph is calculated). VERYLONG: detailed information at every critical point, including the derivatives of the reference field, the evaluation of all other fields, and the flatness. SHELLS: local neighbor environment of every critical point (up to n.i shells, default 10). ESCHER: write the crystal structure description to an octave (escher) script, containing the critical points.

The OBJ keyword writes a <root>_cell.obj file in Wavefront OBJ format. This format can be viewed with view3dscene, meshlab, blender, and many other programs. Alternatively, the PLY and OFF 3d graphics formats can be used. The keywords have the same meaning as in WRITE: MOLMOTIF completes all molecules in the unit cell using atoms from adjacent cells, BORDER includes in the representation the atoms that are exactly on the cell border, CELL represents the unit cell with red sticks, and (ix.i,iy.i,iz.i) tell critic2 to represent that number of cells in each direction. Essentially, the OBJ file is the same as the one generated by WRITE, except in that the critical points are also represented. In the case of a molecular structure (loaded using the MOLECULE instead of the CRYSTAL keyword), the MOLCELL keyword can be used to represent the "molecular cell", that is, the region around the molecule outside which critic2 considers critical points and gradient paths to have diverged to infinity.

9.3 List of properties calculated at points (POINTPROP)

The default output for AUTO contains a list of detailed information at the critical points. This is an example of how this section looks like:

```
* Additional properties at the critical points
[...]
+ Critical point no. 9
  Crystallogrpahic coordinates: 0.65183 0.90869 0.72597
  Cartesian coordinates: 12.89298 22.10026 33.60159
  Type : (3,-1)
  Field value (f): 4.939109713E-03
  Field value, valence (fval): 4.939109713E-03
  Gradient (grad f): -9.5115E-18 3.8415E-17 4.1112E-18
  Gradient norm (|grad f|): 3.978845324E-17
  Laplacian (del2 f): 1.312332087E-02
  Laplacian, valence (del2 fval): 1.312332087E-02
  Hessian eigenvalues: -2.11139E-03 -1.48096E-03 1.67156E-02
  Hessian:
    -2.068242883E-04  -5.185807347E-03   1.179735612E-03
    -5.185807347E-03   1.373314257E-02  -4.510661176E-03
     1.179735612E-03  -4.510661176E-03  -4.029974096E-04
```

For each critical point, the coordinates (Cartesian and crystallographic), the type, and the evaluation of the reference field and its derivatives is given. However, in many cases, it is interesting to calculate the value of a different field, or even an arithmetic expression involving other fields, at those critical points. For instance, it is relatively common to use the kinetic energy density at the bond critical points of the electron density as a measure of bond covalency.

To obtain more information at the critical points of the reference field, the procedure in critic2 is to register a field or an arithmetic expression involving known fields in the "properties list", accesible using the POINTPROP keyword:

```
POINTPROP name.s "expr.s"
POINTPROP shorthand.s
POINTPROP CLEAR
POINTPROP LIST
```

The POINTPROP keyword associates the expression "expr.s" with the name name.s and register that name in a list of properties. When AUTO is run (or CPREPORT, if the POINT-

PROP order comes after AUTO), those arithmetic expression will be applied to each of the CPs and the result printed in the output. For instance, if one does:

```
POINTPROP MYGTF gtf(1)
POINTPROP STH log($1^2)
```

Then the result of AUTO for the critical point above becomes:

```
+ Critical point no. 9
  Crystallogrpahic coordinates: 0.65183 0.90869 0.72597
  Cartesian coordinates: 12.89298 22.10026 33.60159
  Type : (3,-1)
  Field value (f): 4.939109713E-03
  Field value, valence (fval): 4.939109713E-03
  Gradient (grad f): -9.5115E-18 3.8415E-17 4.1112E-18
  Gradient norm (|grad f|): 3.978845324E-17
  Laplacian (del2 f): 1.312332087E-02
  Laplacian, valence (del2 fval): 1.312332087E-02
  Hessian eigenvalues: -2.11139E-03 -1.48096E-03 1.67156E-02
  Hessian:
    -2.068242883E-04  -5.185807347E-03   1.179735612E-03
    -5.185807347E-03   1.373314257E-02  -4.510661176E-03
     1.179735612E-03  -4.510661176E-03  -4.029974096E-04
  mygtf (gtf(0)): 4.112914774E-04
  sth (log($0^2)): -1.062114037E+01
```

The properties in the list are calculated at the end. The properties list is also used in other parts of critic2, notably in the output of POINT (Points (POINT)).

Any arithmetic expression can be used in POINTPROP, but it is common to use on of the chemical functions from the critic2 function library (List of available functions). The shorthand names for the chemical functions can also be used to apply those functions to the reference field. For instance:

```
POINTPROP GTF
```

activates the calculation of the Thomas-Fermi kinetic energy density (gtf function) on the reference field. POINTPROP can only be used with arithmetic expressions involving known fields. The keyword CLEAR deletes all the properties in the list. The list of properties can be accesed at any time using POINTPROP LIST.

10 Graphical representations of gradient paths

10.1 Overview

Gradient paths are the solution of the differential equation x' = grad(f(x)) where f is a scalar field. They play an important role in the QTAIM theory because gradient paths of the electron density can not cross the boundary between atomic regions and, as a consequence, a gradient path plot is a simple way to investigate the shape and properties of a basin. Gradient paths originate at maxima (if the field is the density they are usually the nuclei) and end at the minima (the crystal voids) or at infinity in case of a gas-phase molecule.

There are two keywords for gradient path representation in critic2: GRDVEC (2D gradient paths plus a contour plot) and FLUXPRINT (3D).

10.2 Gradient path representations in a plane (GRDVEC)

```
GRDVEC
   {FILES|ROOT|ONAME} rootname.s
   PLANE x0.r y0.r z0.r x1.r y1.r z1.r x2.r y2.r z2.r
         [SCALE sx.r sy.r]
   PLANE {CP|ATOM} cp1.i [LVEC x.i y.i z.i]
         cp2.i [LVEC x.i y.i z.i] cp3.i [LVEC x.i y.i z.i]
         [SCALE sx.r sy.r] [SIZE zx.r zy.r]
   OUTCP sx.r sy.r
   HMAX hmax.r
   ORIG x.r y.r z.r atr.i up.i down.i
   CP cp.i up.i down.i
   CPALL
   BCPALL up.i down.i
   RBCPALL bup.i bdown.i rup.i rdown.i
   CHECK
        x.r y.r z.r
        ...
   ENDCHECK
   CONTOUR {F|GX|GY|GZ|GMOD|HXX|HXY|HXZ|HYX|HYY|
            HYZ|HZX|HZY|HZZ|LAP} [LIN cini.r cend.r|
            LOG|ATAN|BADER] [nptsu.i nptsv.i niso.i]
ENDGRDVEC
```

GRDVEC plots a plane containing all the gradient paths originating from a set of points. The GRDVEC environment accepts a set of input lines (in any order) that control the characteristics of the plot. The syntax of GRDVEC originated from (and is similar to) Bader's AIMPAC suite of programs.

By using the FILES keyword (equivalently, ROOT or ONAME), the user sets the root name of the output files containing the information for the plot (default: <root>). These files include:

- <root>.grd : gradient path data.
- <root>.dat : values of the field on thea plane.
- <root>.iso, <root>.neg.iso : positive and negative contour lines.
- <root>.gnu : gnuplot script file that generates the merged gradient/contour plot.
- <root>-label.gnu : gnuplot script file loaded in <root>.gnu containing the information for the position of the CPs in the plot plane.

In the standard syntax, PLANE specifies the plane for the plot using three points: x0 is the origin, x1 the end of the x-axis and x2 the end of the y-axis, all in crystallographic coordinates. The SCALE option allows the user to extend the plane relative to the reference points; The x- and y- axis of the rectangle are scaled using the sx.r and sy.r values.

There is another possibility: the PLANE CP keyword (or the equivalent syntax PLANE ATOM) accepts three atoms or critical points from the complete CP (or atom) list. The first atom (or CP) is taken as reference, and the other two are replicated by symmetry until critic2 finds the equivalent position that is nearest to the first. The three points found this way (hopefully) determine a plane. The barycenter of the points is calculated, and is set as the center of the plot window, which consists of a rectangle with sides equal to two times the longest distance from the barycenter to the nuclei. The reference points for the plane in PLANE CP may translated by lattice vectors (optional keyword LVEC).

The plot plane may contain regions that are traversed by gradient lines originating at critical points located outside the plot region. If this is the case, the OUTCP option allows the user to extend the plane when considering which origins to be included. The sx.r and sy.r are scale parameters. The x-axis extends (sx.r-1)*l(x) in each direction, where l(x) is the axis length. The sy.r variable works the same way. Consequently, the plane determined by the vectors given in PLANE acts as a clipping plane while the scaled plane determines the gradient path origins.

With HMAX, you can set the maximum distance from a CP to the plane to be included in the plot. Default: 1d-4.

The ORIG keyword adds a source of gradient lines to the plot. Its crystallographic coordinates are x.r, y.r and z.r. atr.i is 1 if the point is to be treated as a ncp or ccp (the up and down trajectories start from points located on a sphere centered on the origin) and it is 0 if the point is to be treated as a bcp or ccp (a circle is built around the CP in the plane determined by two eigenvectors whose eigenvalues have equal sign. The remaining eigenvector determines a unique direction). up.i and down.i are the number of gradient paths to be started in the upwards and downwards direction respectively.

The CP keyword accepts a critical point identifier from the complete CP list (or the complete atom list). The number of upwards and downwards gradient paths must be given. A special case is the CPALL keyword, which adds as origins every critical point in the CP list on the selected plane. The default number of gradient paths is 36 down for ncps and 36 up for ccps, and 2 up and 2 down for bcps and rcps. The BCPALL keyword is similar to CPALL, except that only the bond critical points are included as origins. If BCPALL is used, the user must supply the number of gradient lines in the upwards and downwards directions. In a similar way, RBCPALL includes bond and ring critical points, and the user must give the number of upwards and downwards gradient paths for bonds (bup.i, bown.i) and rings (rup.i, rdown.i).

The CHECK environment allows the user to enter the crystallographic coordinates of a CP of the scalar field to add it as an origin. If the point given is not a CP or if it lies out of the selected plane, it is ruled out of the origin list. The valid CPs in the CHECK list are identified and an adequate number of gradient paths are started according to its character: for a ncp and ccp, 36 upwards or downwards and for a bcp or rcp, 2 upwards and 2 downwards.

The keywords [F|GX|...] request critic2 to generate a plot in which the gradient paths calculated in GRDVEC are merged with a contour plot, in the spirit of PLANE CONTOUR (see Planes and contour plots (PLANE)). This contour map corresponds to the reference scalar field if F is used, to its Laplacian (LAP), etc. In a way analogous to CONTOUR, the contour lines may be calculated using a linear spacing between isocurves on a remapped scalar field. If LOG is given, log(f) and log(abs(f)) are used while for ATAN, the function is 2/pi*atan(f). The BADER keyword corresponds to a fixed set of iso-values ({1,2,4,8}x10^{-3,-2,-1,0,1}). LEVEL selects a linear set of contour values from cini.r to cend.r. The number of points on each axis of the grid and the number of contour lines can be optionally given with nptsu.i, nptsv.i and niso.i. By default, they are 100, 100 and 100.

Note that GRDVEC is able to handle non-orthogonal axis. If the two plane axis determined in the PLANE keyword are non-orthogonal, the final graph will correctly reflect the actual appearance of the plane by conserving the original angle between the x- and y- axis.

Also, note that at most 2 gradient lines may be traced from bcps and rcps, either upwards or downwards. Thus, for example, BCPALL 2 2 is equivalent to BCPALL 2 100 or BCPALL 100 100.

10.3 Three-dimensional gradient path representations (FLUXPRINT)

```
FLUXPRINT
  POINT {1|-1|0} x.r y.r z.r [step.r epsi.r]
  NCP cp.i ntheta.i nphi.i [step.r epsi.r]
     [LVEC x.i y.i z.i]
  BCP cp.i 1 [step.r epsi.r] [LVEC x.i y.i z.i]
  BCP cp.i {0|-1} n.i [step.r epsi.r]
```

```
      [LVEC x.i y.i z.i]
      [BRAINDEAD|QUOTIENT|DYNAMICAL]
   RCP cp.i -1 [step.r epsi.r] [LVEC x.i y.i z.i]
   RCP cp.i {0|1} n.i [step.r epsi.r]
      [LVEC x.i y.i z.i]
      [BRAINDEAD|QUOTIENT|DYNAMICAL]
   CCP cp.i ntheta.i nphi.i [step.r epsi.r]
      [LVEC x.i y.i z.i]
   GRAPH igraph.i [step.r epsi.r]
   TEXT|TESSEL|TESS|OBJ|PLY|OFF
   EVERY every.i
   SHELLS ishl.i
   NOSYM
 ENDFLUXPRINT
```

The FLUXPRINT keyword prints three-dimensional gradient paths. There are several plotting comands:

- POINT: build a gradient path starting at point (x.r y.r z.r) in cryst. coordinates. step.r is the maximum step (Cartesian coordinates) for the gradient path tracing algorithm. If step.r > 0, use it also as the initial step. If step.r < 0, use a small step as initial (1d-3). epsi.r is the gradient norm stop criterion. The default values are 0.1 for step and 1e-9 for epsi. The {1|-1|0} field controls the direction of the path. An ascending gradient path is obtained with 1 while -1 issues a descending path. 0 = -1 + 1 makes FLUXPRINT represent both ascending and descending paths.

- NCP: print gradient paths starting from a (small) sphere centered on the nuclear CP identified by cp.i (this identifier comes from the complete CP list). The number of points is controlled by ntheta.i (number of points sampling the azimuthal angle) and nphi.i (number of points sampling the polar angle). cp.i specifies a ncp in the main cell up to a lattice translation. The LVEC optional keyword allows the user to enter a lattice vector to displace the represented ncp gradient paths from their position given in the complete cp list written by AUTO.

- BCP: print gradient paths starting at the vicinity of a bond CP, identified by cp.i. If the gradient path is ascending (1 in the fourth field), the (unique) bond path associated to the bcp is represented. If -1 is given instead, the IAS associated to the bcp is sampled starting from a small circle surrounding the bcp, with n.i points on it. With a 0 value, both tasks are performed.

  The three keywords BRAINDEAD, QUOTIENT and DYNAMICAL establish the method employed in generating the starting angular grid. With BRAINDEAD, critic2 uses a uniform angular grid. Using QUOTIENT, the uniform grid is remapped by x^(l1/l2) where l1 and l2 are the two negative eigenvalues at the bcp. This way, the points get accumulated around the bcp with lowest eigenvalue (highest if absolute value is taken). DYNAMICAL uses a linearized model of the interatomic surface and predicts the initial angles critic2 has to take in order to generate a uniform distribution of points a given distance away. This distance is calculated as 90% of the distance to the nearest ccp found in a coarse exploration of the IAS omega-limits. Unfortunately, this algorithm works only in cases where the bcp has significant but not too large ellipticty. Also, there is no gain in using this method in cases where the number of ccps is different than 4.

  By default, BRAINDEAD is used. H1 is experimental.

- RCP: print gradient paths starting at the neighbourhood of a ring CP. The situation is analogous to that of the bcps.

- CCP: print gradient paths starting at the vicinity of a cage CP. Again, the situation is symmetric to the ncp case.

- GRAPH: represent the complete graph in the unit cell. This means:

  - All the bond paths for which both ncps and the bcp lay inside the main unit cell.
  - All the ring paths for which both ccps and the rcp lay inside the main unit cell.

  The critical points on the boundary of the main cell are also represented.

  The igraph.i value represents the quantity of information that is to be printed. It is a sum of values, each representing an element to plot:

  - 1 : print ring paths associated to the rcps.
  - 2 : print bond paths associated to the bcps.

  These options are only available if an AUTO task was carried out with the option SET GRAPH.

Several parameters of FLUXPRINT can be changed using the options keyword. The format of the output file can be controlled with the following keywords: TEXT (plain text file), TESS or TESSEL (tessel), OBJ (Wavefront obj), PLY (ply format), and OFF (Geomview's off). The EVERY keyword makes critic2 write only one in every every.i points for each gradient path.

Finally, the SHELLS keyword applies only to graph and graphcp. It represents the number of unit cell shells where the graph is going to be plotted. Thus, 0 represents the main unit cell; 1, the main unit cell and its 26 neighbours; and so on. By default, SHELLS adopts the -1 value, which is equivalent to 0 for the graph keyword and means that the partial graph generated in graphcp is not expanded through symmetry.

NOSYM: do not use symmetry operations. Using this keyword, the complete list of CPs in the unit cell is written, together with the identity matrix as the only operation in the space group.

11 Atomic basin representations

11.1 Attractor basin plots (BASINPLOT)

```
BASINPLOT [CUBE [lvl.i] | TRIANG [lvl.i] | SPHERE [ntheta.i nphi.i]]
          [OFF|OBJ|PLY|BASIN|DBASIN [npts.i]}]
          [PHASE [phtheta.r phphi.r]] [CP cp.i] [PREC delta.r] [VERBOS
          [MAP id.i|"expr"]
```

Plot the attraction basin of the CP cp.i from the complete list (if CP is not given, all the non-equivalent attractors are used). The rays on which the bisection is carried out are determined by the method chosen. With CUBE, a cube is selected as the starting polyhedron, and recursively subdivided lvl.i times. The final (convex) polyhedron is placed on the attractor and the zero-flux surface limit for the rays is determined. TRIANG follows the same process, starting from an octahedron. SPHERE stands for a direct triangulation of the unit sphere. There are nphi.r parallels. The equatorial circles contain exponentially more points than the polar. Ntheta.i thus represents a seed. The total number of points is given by the formula 2*nphi*(2**ntheta-1)+2.

The output keyword selects the output format for the basin plot: OFF, OBJ, PLY, BASIN, and DBASIN. Note that DBASIN files also contain information about scalar fields measured along the basin rays.

The initial polyhedron may be rotated an angular phase phtheta.r and phphi.r to avoid high symmetry lines.

The naming scheme of the output files is <root>-cp.ext where root is the general root of the run (the name of the input file up to the first dot unless changed by the ROOT keyword), cp is the complete CP list id. of the attractor and ext is the appropriate extension (off, basin or dbasin)

The precision of the bisection is delta.r (set using the PREC keyword). VERBOSE gives more information in the output about the bisection process.

If a 3d model format is used (OFF, OBJ, PLY), the MAP keyword can be utilized to colormap a given field (given by number id.i) or field-containing expression ("expr") onto the surface. The color scale limits are the minimum and the maximum value of the field or expression on all the points of the surface. The mapping function is the same as in gnuplot (r=sqrt(x), g=x^3, b=sin(360*x), with x from 0 to 1).

Default: TRIANG method, lvl.i = 3, ntheta.i = nphi.i = 5, OBJ output, phtheta.r = 0d0, phphi.r = 0d0, all the non-equivalent attractors found in AUTO.

11.2 Primary bundle plots (BUNDLEPLOT)

```
BUNDLEPLOT x.r y.r z.r
```

```
    [CUBE [lvl.i] | TRIANG [lvl.i] | SPHERE [ntheta.i nphi.i]]
    [OFF|OBJ|PLY|BASIN|DBASIN [npts.i]}] [PHASE phtheta.r phphi
    [ROOT root.s] [PREC delta.r] [VERBOSE] [MAP id.i|"expr"]
```

Plot a primary bundle starting from a point in its interior, given by x.r y.r z.r. The bisection algorithm is used with precision delta.r (PREC keyword). The rays traced are obtained by a recursive subdivision (lvl.i cycles) a cube (CUBE), an octahedron (TRIANG) or using a uniform distribution of ntheta.i * nphi.i points on the unit sphere (SPHERE). The output file has root root.s, and its format may be OFF, OBJ, PLY, BASIN or DBASIN with npts.i points sampled along each ray. The initial polyhedron may be rotated a phase given by phtheta.r (polar angle) and phphi.r (azimuthal angle).

If a 3d model format is used (OFF, OBJ, PLY), the MAP keyword can be utilized to colormap a given field (given by number id.i) or field-containing expression ("expr") onto the surface. The color scale limits are the minimum and the maximum value of the field or expression on all the points of the surface. The mapping function is the same as in gnuplot (r=sqrt(x), g=x^3, b=sin(360*x), with x from 0 to 1).

Default values: TRIANG, lvl.i = 3, ntheta.i = nphi.i = 5, OBJ output, phtheta.r = 0d0, phphi.r = 0d0, root.s = <root>-bundle.

12 Integration of atomic basins

12.1 Overview

Critic2 provides several methods to integrate the attractor basins associated to the maxima of a field. In QTAIM theory, this field is the electron density, the attractors are (usually) nuclei and the basins are the atomic regions. In this case, the integrated properties are atomic properties (e.g. atomic charges, volumes, moments, etc.). The attractor basins are defined by a zero-flux condition of the electron density gradient: no gradient paths cross the boundary between attractor regions. This makes the basins local to each attractor, but their definition yields a relatively complex algorithmic problem.

The simplest way of integrating an attractor basin is bisection. A number of points distributed in a small sphere around the atom are chosen, each of them determining a ray. On each ray, a process of bisection is started. A point belongs to the basin if the gradient path traced upwards ends up at the position of the attractor we are considering. If the end-point is a different attractor, then the point is not in the basin. By using bisection, it is possible to determine the basin limit (called the interatomic surface, IAS). The bisection algorithm is implemented in critic2, and can be accessed with the INTEGRALS keyword.

An algorithm has been proposed based on the recursive subdivision of the irreducible Wigner-Seitz (IWS) cell, called qtree. In qtree, the smallest symmetry-irreducible portion of space is considered and a tetrahedral mesh of points is superimposed on it. The gradient path is traced from all those points and the points are assigned to different atoms (the points are 'colored'). The final integration is performed by quadrature. The qtree algorithm is accessed through the QTREE keyword.

Lastly, integration algorithms based on grid discretization are very popular nowadays thanks to the widespread use of pseudopotential/plane-waves DFT methods. Critic2 provides the integration method of Yu and Trinkle (YT), described in JCP 134 (2011) 064111. The algorithm is based on the assignment of integration weights to each point in the numerical grid by evaluating the flow of the gradient using the neighboring points. This algorithm is extremely efficient and robust and is strongly recommended in the case of fields on a grid. The keyword is YT. Another alternative for grids is the method proposed by Henkelman et al. (Comput. Mater. Sci. 36, 254-360 (2006), J. Comput. Chem. 28, 899-908 (2007), J. Phys.: Condens. Matter 21, 084204 (2009)), which is implemented in critic2 using the keyword BADER.

The field that determines the basins being calculated is always the reference field (see The reference field). However, it is in general interesting to one or more integrable properties using other scalar fields. For instance, we can calculate the charge inside an ELF basin: the ELF would be the reference field and the electron density would be an integrable property.

12.2 List of properties integrated in the attractor basins (INTEGRABLE)

```
INTEGRABLE id.i [F|FVAL|GMOD|LAP|LAPVAL|
            {MULTIPOLE|MULTIPOLES} [lmax.i]] [NAME name.s]
INTEGRABLE "expr.s"
INTEGRABLE CLEAR
```

The list of integrable properties follows a similar strategy to field loading (Loading a field (LOAD)) and the list of properties calculated at the critical points (List of properties calculated at points (POINTPROP)). There is an internal list of the properties that will be integrated in the attraction basins, which can be modified by the user using the INTEGRABLE keyword.

INTEGRABLE defines a new quantity to be integrated in the basins. The new integrable property is related to field number id.i as an integrable property. This quantity can be the field value itself (F), its valence component (if the field is core-augmented, FVAL), the gradient norm (GMOD), the Laplacian (LAP), or the valence-component of the Laplacian (LAPVAL). By default, F is used. With MULTIPOLES (or MULTIPOLE), the multipole moments of the field are calculated up to l=lmax.i (default: 5). This keyword only applies to the BADER and YT integration methods; for the others, it is equivalent to the field value (same as F). The name for the integrand can be changed using the keyword NAME.

In addition, it is possible to define an integrable property using an expression involving more than one field (expr.s). For instance, if the spin-up density is in field 1 and the spin-down density is in field 2, the atomic moments can be obtained using:

```
LOAD AS "$1+$2"
REFERENCE 3
INTEGRABLE "$1-$2"
```

The default integrable properties are:

- Volume (1)
- Charge (f): the value of the reference field is integrated. If the reference field is the density, then it is indeed the charge. Otherwise, the 'charge' in the output is only a label for the integrated field on the basin.
- Laplacian (lap(f)): the Laplacian of the reference field. The integrated Laplacian has been traditionally used as a check of the quality of the integration because the exact integral is zero regardless of the basin (because of the divergence theorem). However, it is difficult to obtain a zero in the Laplacian integral in critic2 because of numerical inaccuracies:
  - In fields based on a grid, the numerical interpolation gives a noisy Laplacian.
  - In FPLAPW fields (WIEN2k and elk), the discontinuity at the muffin surface introduce a non-zero contribution to the integral.

The keyword CLEAR resets the list to its initial state (volume, charge and Laplacian). Using the INTEGRABLE keyword will print a report on the list of integrable properties.

12.3 Bisection

```
INTEGRALS {GAULEG ntheta.i nphi.i|LEBEDEV nleb.i}
          [CP ncp.i] [RWINT] [PHASETH theta.r]
          [PHASEPH phi.r] [VERBOSE]
```

Integrate the attractor basins using bisection. Ntheta.i and nphi.i are the number of theta (polar angle) and phi (azimuthal angle) points for the Gauss-Legendre quadrature, if GAULEG is used. The number of azimuthal angles depends on the actual value of the polar angle (theta) and is adapted according to the formula:

realnphi = int(nphi.i * sin(theta)) + 1

In the case of a Lebedev-Laikov quadrature, the number of points of the radial Gauss-Legendre grid and the octahedral grid is needed. The actual value of nleb.i is the smallest number larger than the one given by the user included in the list: 6, 14, 26, 38, 50, 74, 86, 110, 146, 170, 194, 230, 266, 302, 350, 434, 590, 770, 974, 1202, 1454, 1730, 2030, 2354, 2702, 3074, 3470, 3890, 4334, 4802, 5294, 5810.

By using the CP keyword, a single non-equivalent CP (ncp.i) is integrated. Otherwise, all the CPs of the correct type (found using AUTO) are integrated. If RWINT is present, read (if they exist) and write the .int files containing the interatomic surface limit for the rays associated to the chosen quadrature method. The PHASETH and PHASEPH keywords are used to rotate the sphere quadrature (for example, to avoid symmetry lines).

Defaults: ntheta.i = nphi.i = 50, nleb.i = 4802.

```
SPHEREINTEGRALS {GAULEG ntheta.i nphi.i|
                 LEBEDEV nleb.i} [CP ncp.i] [NR npts.i]
                [R0 r0.r] [REND rend.r]
```

Integrates the volume, field and Laplacian in successive spheres centered around each of the attractor CPs. The same considerations for GAULEG and LEBEDEV as in the keyword above apply.

A total number of npts.i spheres are integrated per nucleus. The grid is logarithmic, so that the region near the nucleus has a higher population of points. The grid starts at the radius r0.r and ends at rend.r . If rend.r < 0 then the final radius is taken as half the nearest neighbor distance for each atom times abs(rend.r).

Default: npts.i = 100. In GAULEG, ntheta.i = 20 and nphi.i = 20. In LEBEDEV, nquad.i = 770. r0 = 1d-3. rend = rnn/2 for each CP. id.i = 0 (all attractors).

12.4 Qtree

```
QTREE_MINL minl.i
GRADIENT_MODE gmode.i
QTREE_ODE_MODE omode.i
STEPSIZE step.r
ODE_ABSERR abserr.r
INTEG_MODE level.i imode.i
INTEG_SCHEME ischeme.i
KEASTNUM k.i
PLOT_MODE plmode.i
PROP_MODE prmode.i
MPSTEP inistep.i
QTREEFAC f.r
CUB_ABS abs.r
CUB_REL rel.r
CUB_MPTS mpts.i
AUTOSPH {1|2}
SPHFACTOR {ncp.i fac.r|at.s fac.r}
```

```
SPHINTFACTOR atom.i fac.r
DOCONTACTS
NOCONTACTS
ACTIVE atom.i
INACTIVE atom.i
WS_ORIGIN x.r y.r z.r
WS_SCALE scale.r
WS_EPS_VOL eps_vol.r
NOWS
KILLEXT
NOKILLEXT
CHECKBETA
NOCHECKBETA
PLOTSTICKS
NOPLOTSTICKS
COLOR_ALLOCATE {0|1}
SETSPH_LVL lvl.i
VCUTOFF vcutoff.r
QTREE maxlevel.i plevel.i
```

The QTREE integration method is a new algorithm capable of calculating the QTAIM atomic properties in a more efficient way than the bisection approach. QTREE is specific to solid-state problems, and is based on a hierarchical subdivision of the irreducible part of the WS cell, employing a tetrahedral grid. The integration region is selected so as to maximize the use of symmetry, and partitioned into tetrahedra. These tetrahedra enter a recursive subdivision process in which each of them is divided in 8 at each level, up to a level given by the user, the maxlevel.i indicated after the QTREE keyword (default, 6). Every tetrahedron vertex is assigned to a non-equivalent atom in the unit cell by tracing a gradient path. Finally, the tetrahedra are integrated and the properties assigned to the corresponding atoms. The space near the atoms is integrated using a beta-sphere, a method that proves more accurate despite generating a second interface in the atomic basin.

In the simplest approach, qtree can be executed using:

```
QTREE [maxlevel.i]
```

where maxlevel.i is the level of subdivision. The optional plevel.i value corresponds to the pre-splitting level of the tetrahedra. The initial tetrahedra list is split into smaller tetrahedra plevel.i times. This can be useful in cases where a very high accuracy (and therefore a very high level) is required, but there is not enough memory available to advance to higher maxlevel.i. However, using plevel.i incurs in a overhead, because the painting procedure is not as efficient when smaller tetrahedra are used.

Note: parallelization might not work with older versions of ifort. The newer versions of gfortran do work with parallel qtree.

The steps of the algorithm are:

- The WS cell is constructed and split into tetrahedra, all of which have in common, at least, the origin. Then, the site symmetry of the origin is calculated and the tetrahedra that are unique under the operations of this group are found. This is what we call the irreducible Wigner-Seitz cell (IWS). Note, however, that it is only 'irreducible' in the local symmetry of the origin, not by the full set of space group operations. The IWS is the region that is to be integrated in later steps of QTREE. We will refer to single IWS tetrahedra as IWST.

  It is possible, through the WS_ORIGIN keyword, to shift the origin of the WS cell away from the (0 0 0) position. Using the procedure above, the number and shape of IWST changes depending on the origin chosen. Trivially, a general position will make the IWS exactly equal to the WS.

  Also, for large systems, the user can choose to shrink the size of the original WS cell in order to integrate a smaller region, using the WS_SCALE keyword (most likely in combination with WS_ORIGIN, to move the region around). If a value is given to WS_SCALE (say rws), then all the vectors connecting the origin of the WS cell with the vertex are shrunk by a factor rws, therefore decreasing the volume by a factor rws**3 . The IWS is calculated using the smaller WS cell and integrated in the same way. Note that this integration region is non-periodic: it does not fill the volume of the solid and it does not integrate to the total number of electrons per cell.

- Non-overlapping spheres are chosen centered on each of the atoms of the cell, the so-called beta-spheres. Atoms equivalent by symmetry share the same beta-sphere radius (say beta_i for atom i). The beta-sphere takes two roles in QTREE:

  - The atomic properties are integrated inside the beta-spheres using a 2d cubature. The cubature can be a product of two 1d Gauss-Legendre quadratures or a Lebedev quadrature of the sphere. Both methods, and the number of nodes can be selected using the INT_SPHEREQUAD_* keywords explained below. The radial quadrature can be any of the available in critic, and is controlled by the INT_RADQUAD_* options. The default values, however, are usually fine, integrating the beta-spheres in a matter of seconds with a precision that is orders of magnitude better than the overall QTREE performance.

    This beta-sphere integration removes the error of the finite-elements integration of a region where the integrated scalar fields present the steeper variations in value. By removing the high-error regions from the grid integration, the accuracy of QTREE is enhanced. In particular, this increase in precision outweighs the loss by creating an additional interface between the grid and the sphere.

  - The space inside the beta-sphere of an atom is assumed to be inside the basin of that atom. The terminus of any gradient path that reaches the interior of the beta-sphere i is assumed to be the atom i. It is known that most of the steps in the integration of the gradient of the electron density are spent in the close vicinity of the terminus. Therefore, this modification saves precious function evaluations.

The default beta-sphere radius is set to 0.80 times half the nearest-neighbor distance. Both i) and ii) above assume that the beta-sphere is completely contained inside the basin of the atom. This may turn out not to be true for the default beta-sphere radius (specially for cations in ionic systems). In these cases, the keyword SPHFACTOR is used. Its syntax is:

```
SPHFACTOR 1 0.70    ! Make b_1 = 0.70 * rNN2(1) (atom type 1)
SPHFACTOR 0 0.60    ! Make b_i = 0.60 * rNN2(i) for all atoms
SPHFACTOR Si 0.60   ! Make b_i = 0.60 for all Si atoms
```

If SPHFACTOR < 0, use the scheme by Rodriguez et al. to determine the beta-sphere radii (JCC, 30 (2009) 1082-1092): a collection of points around the atom are selected and the angle between the gradient and the radial direction is determined. If all the angles are $<$ 45 degrees, the sphere is accepted. In solids, this strategy yields usually spheres that too large.

In the case that any atomic SPHFACTOR is zero (the default value for all atoms), then a pre-computation at a lower level is done to ensure that all beta-spheres lie within the desired basins. There are two methods for this that can be chosen using AUTOSPH (default: method number 2).

Method number one involves a reduced version of QTREE. The pre-computation usually takes no longer than some minutes (and usually only few seconds) and the spheres are guaranteed to be inside the basins. The keyword SETSPH_LVL controls the level of the pre-computation, that must not be higher than 7. The default value is 6 or maxl, whichever is smaller.

The second method (default) traces gradient path on a coarse sphere around each nucleus, and reduces the sphere until all of the points are inside the basin. NOCHECKBETA is used in this case.

An additional factor the user can define is the SPHINTFACTOR. It is possible to consider the sphere where GPs terminate different from the one that is integrated. If SPHINTFACTOR is defined, as in:

```
SPHINTFACTOR 1 0.75
```

then the sphere associated to atom 1 where the integration is done has a radius which is 0.75 times that of the sphere where GP terminate.

The CHECKBETA and NOCHECKBETA keywords activate and deactivate the check that ensures that the beta-spheres is completely contained inside the basin.

If a beta-sphere is not strictly contained in the basin, QTREE detects it and stops immediately (specifically, QTREE checks that every tetrahedron that is partly contained in a beta-sphere has vertex termini that are all assigned to the same atom as the beta-sphere owner).

For a new system, it is always a good idea to start with a low-level QTREE (say, level 4) to check if the default beta-spheres are adequate. If one of the beta-spheres is too large, the error message looks like:

```
An undecided tetrahedron is overlapping with a beta-sphere.
Make beta-spheres smaller for this system.
terms:             1          -1           2           1
```

which indicates that there is a tetrahedron that is partly contained in the sphere of the first atom (terminus -1) and that has a vertex corresponding to atom 2. Modifying the sphfactor solves this problem:

```
SPHFACTOR 1 0.70
```

At lower levels, QTREE is reasonably fast, so a trial-and-error selection of beta-spheres is acceptable.

Note that the beta-spheres used in QTREE have no relation to the ones reported after an AUTO calculation.

- If the cell is periodic (which means that WS_SCALE was not set), the contacts between the faces of the IWST are found. These contacts are used in a later step to copy the termini information between tetrahedron faces.

  The determination of the tetrahedra contacts in a periodic integration region is deactivated if the NOCONTACTS keyword is issued. The opposite is the DOCONTACTS keyword. By default, the contacts are not calculated.

- A grid is built for each of the IWST, for which the termini of the gradient paths starting at each of the grid points will be calculated. The grid is determined by subdividing the IWST qtree_lvl times. In each subdivision step, a parent tetrahedron is divided in 8 smaller tetrahedron (all with the same volume, V / 8) by splitting each edge of the parent tetrahedron in two. There are two possible ways of doing this, the election being irrelevant to the performance of QTREE.

  For a given IWST, the size of the grid is given by S = n*(n+1)*(n+2)/6 where n = 2**qtree_lvl + 1, the approximate scaling being as 8**qtree_lvl . The termini information on the grid is saved to the array trm of type integer*1, with size nt * S, where nt is the number of IWST (in fact, the integer type is that which is the result of selected_int_kind(2)).

The subdivision level is the main input parameter for QTREE, controlling the accuracy (and cost!) of the integration. For small-medium sized systems 4-5 are low cost integrations (seconds), 6-7 are medium cost (minutes) and 8-9 are the slowest and most accurate (hours). The level is input in the call to the QTREE integration:

```
QTREE 6   ! qtree_lvl is 6
```

By default, the integration level is 5.

In addition to trm, more work space can be allocated if the integration is restricted to the volume and charge or to the volume, charge and Laplacian. The number and type of properties to be integrated is controlled by the PROP_MODE keyword. The following values are allowed:

- 0 - only volume is integrated. This amounts to canceling the finite elements integration of tetrahedra and is equivalent to INTEG_MODE 0 (see below).
- 1 - only charge and volume. If the integration uses the value of the density at the grid points (INTEG_MODE 11, see below) In addition to trm, another real*8 array, fgr, is allocated (strictly it is selected_real_kind(14), not real*8). In fgr, the value of the density at the grid points is stored.
- 2 - charge, volume and Laplacian. In a similar way to 1, if the information on the grid points is used during the integration (INTEG_MODE 11), an additional real*8 array, lapgr, is allocated. It contains the value of the Laplacian of the electron density at the grid points.
- 3 - all the integrable properties calculated by the module. The number of properties varies with the interface being used. No fgr or lapgr are allocated, as the grid points need to be recomputed during integration.

The default value for PROP_MODE is 2.

The termini of the grid points contained in a beta-sphere is marked previous to the beginning of the subdivision.

- Each tetrahedron is subdivided recursively up to a level qtree_lvl, and integrated at the same time. The IWST integration is relatively independent of one another, so for the moment we will focus on just one IWST, which we will call the base tetrahedron.

  The IWST integration is not exactly independent of one another for two reasons:

  - When the integration ends, the termini of the four faces of a base tetrahedron are copied to its neighbors' trm, according to the contacts determined previously.
  - Depending of the method chosen (see GRADIENT_MODE below), the gradient path integration may be aware of the neighboring grid points, that may very well

belong to other IWST. In particular, the gradient mode number 3 integrates a gradient path following grid points. When the endpoint is reached, all the grid points that have been traversed by the path are assigned the same common terminus. Therefore, there is the possibility that gradient paths starting inside a given base tetrahedron write the trm of other IWST.

Nevertheless, both features can be avoided if there is ever interest in parallelizing the integration over IWST.

- A tetrahedra stack is built and initialized only one element: the base tetrahedron. An iterator works on the stack, performing at each step the following tasks:
  - Pop a tetrahedron from the stack.
  - The termini of the vertex of the tetrahedron are calculated, if they are not already known. Let us assume for now that we have a method that traces a gradient path and reliably locates the terminus for a given grid. This will be treated below.
  - If all the termini of the tetrahedron correspond to the same atom, its inside of the tetrahedron is 'painted'. This means that all the grid points that are in the interior or border of the tetrahedron are assigned the same color as its vertex, thereby saving the tracing of the gradient paths.

    This 'painting' can be dangerous whenever a (curved) IAS crosses the face of the tetrahedron which undergoes the operation. To this end, a minimum level is defined, using the keyword QTREE_MINL. If the subdivision level of the tetrahedron is lower or equal than QTREE_MINL, the tetrahedron is not painted. (Note: the base tetrahedron corresponds to level 0).

    Furthermore, if all the termini correspond to the same atom and are located outside of the beta-sphere region, the tetrahedron is integrated and does not enter another subdivision process. Once more, this only happens to tetrahedra with a level of subdivision strictly greater than QTREE_MINL. As we did with the gradient path tracing, let us suppose that we have at our disposal a method that calculates the integral of the selected properties over one of these tetrahedra. We will refer to this methods as an 'inner integration method', because the value of the properties will be assigned to only one atom.

    On the contrary, if all the termini are located inside the beta-sphere region, the tetrahedron does not subdivide, but the properties are not integrated, because this region corresponds to the sphere integration addressed in point 2.

    If the tetrahedron is in the border of a beta-sphere, it is divided further.
  - If the tetrahedron is at subdivision level equal to qtree_lvl, then it does not subdivide, it is integrated and the properties are assigned to the atoms. There are several possibilities depending of the nature of its vertex termini:

        * If all the termini correspond to the same atom, and the tetrahedron is completely inside or outside of this atom's sphere, it corresponds to a case in 6.3.
        * If it is completely inside an atom basin, but on the border of a beta-sphere (remember that being on the border of a beta-sphere implies that it is inside the basin), the part of the tetrahedron that is outside of the sphere is integrated and assigned to the atom. Another integration method is required for this, essentially different from the 'inner integration'. We will refer to this one as 'border, same-color integration'.
        * If the tetrahedron has termini corresponding to different atoms, its properties are integrated and split into contribution to atoms, according to the number of termini each atom has. These tetrahedra are located on the IAS, and require a third class of integration, 'border, diff-color integration'.
    - A tetrahedron that has not been integrated continues to the subdivision step. In this step, 8 new tetrahedra are pushed into the stack. To this end, the edges of the parent tetrahedron are split in two. Note that, by construction, the newly generated points correspond to grid points also.
      The subdivision scheme is:
        * 1, 1-2, 1-3, 1-4
        * 2, 1-2, 2-3, 2-4
        * 3, 1-3, 2-3, 3-4
        * 4, 1-4, 2-4, 3-4
        * 2-3, 1-2, 1-3, 1-4
        * 1-4, 1-2, 2-3, 2-4
        * 1-4, 1-3, 2-3, 3-4
        * 2-3, 1-4, 2-4, 3-4

      where 'a' represents a vertex of the parent tetrahedron and 'a-b' the midpoint of both vertex. Each of the 8 child tetrahedra enclose the same volume, equal to V / 8**l, where l is the subdivision level and V is the volume of the base tetrahedron. The 4 last subdivisions can be chosen in yet another way, but this is not relevant to the results of QTREE.
    - When the stack is empty, the work on the base tetrahedron is finished.

- 'Inner integration'. The inner integration applies to tetrahedra that are completely contained in the non-beta-sphere region of a basin. It can apply to a tetrahedron of any level, as long as it is greater than QTREE_MINL. The integrated properties are to be assigned to a single atom, so the only problem with inner integration is to obtain an accurate value of these integrals.

  In the current implementation of QTREE, several integration methods are possible, and are controlled by the INTEG_MODE keyword. The possible values of INTEG_MODE are:

- 11 : use the information of the density, Laplacian and properties at the vertex of the tetrahedron to integrate. The integral is approximated by a quadrature of four terms, each corresponding to a volume that is 1/4 of the volume of the tetrahedron and multiplied by the value of the properties at the vertex. This integration method is useful if only charge or charge and Laplacian are being integrated, because the information gained during the gradient path tracing, and saved in the fgr and lapgr arrays, is used. Nevertheless, it is not very accurate for large tetrahedra.
- 12 : use the CUBPACK routines. CUBPACK provides an adaptive tetrahedron integration method based on recursive subdivision (exactly the same as QTREE, by the way) and an integration rule with 43 nodes (degree 8), that is equivalent to the DCUTET library by Bernsten et al. The integration rule is fully symmetric under the Th group operations. The error estimation is compared to the error requested by the user, that is controlled using the CUB_ABS (absolute error), CUB_REL (relative error) and CUB_MPTS (maximum number of function evaluations) keywords. If CUB_MPTS is exceeded, an error message is output, but the QTREE integration continues.

  Note that, no matter how low the error requirements are, the CUBPACK integration spends, at least, 43 function evaluations per tetrahedron, so it is quite expensive if compared to other integration modes. This should be reserved for large tetrahedra (see below) or for really accurate calculations.
- 1...10 : use a non-adaptive rule from the KEAST library (Keast et al., 1986), the number corresponding to:
  - 1 -- order = 1, degree = 0
  - 2 -- order = 4, degree = 1
  - 3 -- order = 5, degree = 2
  - 4 -- order = 10, degree = 3
  - 5 -- order = 11, degree = 4
  - 6 -- order = 14, degree = 4
  - 7 -- order = 15, degree = 5
  - 8 -- order = 24, degree = 6
  - 9 -- order = 31, degree = 7
  - 10 -- order = 45, degree = 8

  In particular, the first KEAST rule uses the barycenter of the tetrahedron.

The syntax of the INTEG_MODE keyword is:

INTEG_MODE lvl.i mode.i

where mode.i is one of the modes above and lvl.i is the level for which it applies. This means that, if a tetrahedron of a given level is to be integrated, the value of INTEG_MODE(level) is checked to decide on the method.

A last INTEG_MODE value is possible:

- -1 : do not integrate and force the tetrahedron into the subdivision process. This value of INTEG_MODE can be combined with a positive value at higher levels, amounting to a recursive integration in the style of CUBPACK. Of course, -1 is not an acceptable value for the last level, qtree_lvl .

As setting these INTEG_MODE by hand could be confusing, QTREE provides some more-or-less well tested sets of INTEG_MODE values, which we will call integration schemes. An integration scheme conveys a full set of INTEG_MODEs that span from the lowest to the highest level of subdivision. Integration schemes are selected using the INTEG_SCHEME keyword, that can assume the following values:

- 0 : do not integrate, only calculate volume and plot (see below). This is equivalent to setting PROP_MODE to 0.
- 1 : subdivide each tetrahedron up to the highest level and then integrate using the vertex information. This is most useful if PROP_MODE is 1 (only charge and volume) or 2 (charge, volume and Laplacian) because the information of the gradient path tracing (fgr and lapgr arrays) are used. INTEG_MODE = -1 -1 ... -1 11 ! ^ ^ ^ ^ ! QTREE_MINL qtree_lvl
- 2 : subdivide each tetrahedron up to the highest level and then integrate using the barycenter. INTEG_MODE = -1 -1 ... -1 1.
- 3 : barycentric integration at all levels of subdivision. Less accurate but faster than 2. INTEG_MODE = 1 1 ... 1 1.
- 4 : one of the Keast rules (given by the KEASTNUM keyword) is used at all levels. If KEASTNUM is n, INTEG_MODE = n n ... n n.
- 5 : CUBPACK, at all levels. Reserve this one for special occasions. INTEG_MODE = 12 12 ... 12 12.
- 6 : this scheme and the next are (poor) attempts at trying an adaptive integration scheme. They are not more reliable or efficient than, say, the scheme 2. Integration scheme 6 calculates levels 4, 5 and 6 using CUBPACK, and the rest with subdivision up to the highest level and vertex-based integration. INTEG_MODE = 12 12 12 -1 ... -1 11.
- 7 : same as 6 but the final integration is based on the barycenter. INTEG_MODE = 12 12 12 -1 ... -1 1.
- -1 : let the user enter the INTEG_MODEs by hand.

The default integration scheme is 2, suitable for low and medium-accuracy calculations.

- 'Border, same-color integration'. This integration applies to tetrahedra that have reached the maximum subdivision level and sit on the interface between a beta-sphere and the atomic basin. Some of the vertex are known to be inside the sphere and some of them are out. The objective is to integrate the out-of-sphere part and summing it to the atomic properties, while ignoring the in-sphere part (that has been integrated at the beginning using a sphere cubature).

  The integration follows by assuming that the sphere radius is much larger than the tetrahedron characteristic lengths and, therefore, that the sphere surface can be considered a plane that intersects the tetrahedron. The intersection points of the sphere with the tetrahedron edges are easily calculated and, for the sake of simplicity in the explanation, we will refer to them as the 'middle' of the edges. Note, however, that in the implementation, these points are calculated exactly. There are three possible cases:

  - One vertex is outside, three inside. The tetrahedron formed by the vertex that is outside and the three middle points of the edges that stem for it form a tetrahedron by itself, that is integrated and added to the atom properties.
  - Three vertex are outside, one inside. The difference between the whole tetrahedron integration and the small tetrahedron inside the sphere is added to the atom properties. The small tetrahedron is formed by the vertex that is inside the sphere and the three edges connected to it.
  - Two vertex are inside, two outside. The region outside of the sphere is a 'triangular prism', that is split in three tetrahedra and integrated.

  Note that the INTEG_MODE of the maximum subdivision level (qtree_lvl) applies to all the sub-integrations of the border, same-color integration.

- 'Border, diff-color integration'. As in the case of 'border, same-color integration', this method only applies to tetrahedra which are at their maximum subdivision level. In this case, the termini of the vertex corresponding to, at least, two different atoms.

  In the current implementation of QTREE, the tetrahedron is integrated as a whole. Then, the properties are equitatively assigned to each of the termini atoms. For instance, if the termini are (1 1 1 3), the properties of the tetrahedron are integrated, then 3/4 of them assigned to atom 1 and 1/4 to atom 3.

  In the literature, this problem has been addressed (although in cubes, not in tetrahedra) by using a Monte-Carlo integration inside the region. This requires tracing a gradient path for each of the random points, to achieve a overall accuracy that scales as sqrt(N) where N is the number of nodes. However, I feel that, if this type of approach is to be used, then it is better to continue subdividing the tetrahedron, in a way that is equivalent to going to a higher QTREE level, that scales as N.

- Gradient path tracing. The gradient path start always at grid points, and are traced using one of three methods, controlled by the GRADIENT_MODE keyword, that can assume the following values:
  - 1 : 'full gradient'. This method is ODE integration as it is meant to be. The integration is carried out ignoring the grid information. As was explained before, the gradient path is terminated whenever it enters a beta-sphere region.
  - 2 : 'color gradient'. At each point of the gradient path, the neighboring grid points are checked. If all of them correspond to the same atom, then the terminus of the gradient path is assigned to that atom.

    In a tetrahedral mesh, the meaning of 'neighboring grid points' is not as clear as in a cubic mesh. For a given point x, the neighbors are calculated by first converting x to convex coordinates, that range from 0 to 2**l (restricted to x_1 + x_2 + x_3 <= 2**l). The neighboring points are:

    ```
    (x_1 +- 1, x_2 +- 1, x_3 +- 1)
    ```

    If any of these neighbors are not valid points in the tetrahedron, they are discarded and not checked. This is the default, except in the grid module.
  - 3 : 'qtree gradient'. This method behaves much like the 'full gradient', but whenever the gradient path steps near a grid point, it is projected to it. When a projection occurs, the grid point is pushed into a stack. At the end of the gradient path, when the terminus is known, all the grid points in the stack are popped and assigned the terminus.

    The projection regions are spheres located around each grid point, whose radius is controlled by the QTREEFAC keyword. The radius of these spheres is minlen / 2**qtreelvl / qtreefac , where minlen is the smallest edge length of the full set of IWST and qtreelvl is the maximum subdivision level. Note that QTREEFAC equals 1 is the maximum value allowed, and corresponds to touching spheres along, at least, one tetrahedron edge. By default, QTREEFAC is 2, that is a compromise value. Lower levels of QTREEFAC tend to give errors when assigning the grid points that lie on the IAS of two atoms (although *only* there). With higher levels, the time saving is gone, and 'qtree gradient' reduces to 'full gradient'.

    Additionally, the projection can be started only after a certain number of initial steps. The MPSTEP keyword controls this value, and defaults to 0.
  - -1, -2, -3: these correspond to the same as their positive values, but each gradient path terminus is compared to their 'full gradient' version, using the best available ODE integration method. Information about the results of the comparison is output to stdout, and a .tess file is generated (difftermxx.tess, where xx is the subdivision level) containing the position of the points where both termini differ.

If the integration region is not periodic, then methods 'color' and 'qtree' are not defined. There are two possible options, controlled by the 'KILLEXT' and 'NOKILLEXT' keywords. If KILLEXT is active (the default behavior), the gradient path tracing is killed whenever it leaves the integration region, independently of the GRADIENT_MODE being used. The terminus is then assigned to an 'unknown' state, and the tetrahedra it generates are not integrated. If NOKILLEXT is active, the gradient path is continued as a 'full gradient', until the terminus is found.

The default is KILLEXT because, if the integration region is not periodic, the integral over atoms that are partially contained in it is most likely not meaningful to the user.

The ODE integration method can be chosen using the QTREE_ODE_MODE keyword, that can assume the following values:

- 1 : Euler method, fixed step, 1st order.
- 2 : Heun method, fixed step, 2nd order.
- 3 : Kutta method, fixed step, 3rd order.
- 4 : Runge-Kutta method, fixed step, 4th order.
- 5 : Euler-Heun embedded method, adaptive step. 1st order with 2nd order error estimation. 2 evaluations per step.
- 6 : Bogacki-Shampine embedded method, adaptive step. 3rd order with 5th order error estimation. The FSAL (first step also last) allows only 4 evaluations per step. Local extrapolation.
- 7 : Runge-Kutta Cash-Karp embedded method, adaptive step. 4th order with 5th order error estimation. 6 evaluations per step.
- 8 : Dormand-Prince 4-5 embedded method, adaptive step. 4th order with 5th order error estimation. 6 evaluation per step, with FSAL. Local extrapolation.

For embedded methods (4--8), the absolute error requested to the method can be set using the ODE_ABSERR keyword. The default of this variable is chosen so that reasonable stepsizes are kept. This default is 1d-3 for Euler-Heun and 1d-4 for the rest. Note that it makes no sense an equivalent ODE_RELERR keyword.

Experimentally, using a method with n more evaluations is better than reducing the step size of the lower accuracy method n times. Additionally, there is no upper limit to the step size (I am assuming nobody is going to use ODE_ABSERR = 1d2), so methods with greater accuracy (7 and 8) save evaluations by increasing stepsize to values much larger than their lower accuracy counterparts.

The step size of the fixed step methods (1--4) is controlled with the STEPSIZE keyword. In the variable step methods (5--8) the value of STEPSIZE is used as the starting step.

The default QTREE_ODE_MODE is Dormand-Prince (8).

- When the integration of the base tetrahedron is finished, the termini of the grid points located at each one of its four faces are copied to the corresponding neighboring IWST using the information found in section 3. Of course, this is only done if DOCONTACTS is active.

- Once the integration of the IWST is completed, the atomic properties are scaled and summed to the integrals inside the beta-spheres. The final result is output, together with an analysis of the contribution of each subdivision level to the total integrated properties.

- It is possible to plot the basins obtained by QTREE using the PLOT_MODE keyword. It can assume the values:

  - 0 : no plotting is done.
  - 1 : a single tess file is written containing a description of the unit cell CPs, the IWS, and balls corresponding to all the grid points that have been sampled.
  - 2 : same as 1 but only the balls that are either on the face of an IWST or close to a IAS are output.
  - 3 : the full WS cell
  - 4 : a file for the full WS cell and several files, containing a description of each of the integrated basins. Note that the basins need not be connected.
  - 5 : same as 4 but only balls belonging to faces of IWST and IAS are output.

  The default value is 0. If PLOT_MODE is >0, then it is possible (and it is active by default) to plot the sticks that form the tetrahedra inside .stick files. The PLOTSTICKS and NOPLOTSTICKS keywords control this behaviour.

Some additional considerations:

- The integration of the volume is not done using the beta-sphere / basin separation because the volume of each tetrahedron is exactly known. This implies that the integrated cell volume for a periodic integration region will always be exact (if it is not, then it is an error). The integrated cell charge, on the contrary, is a measure of how well the tetrahedra are being integrated, but *not* of how well the IAS is being determined.

- For very high levels of QTREE (say 10--11, depending on the amount of memory your computer has), memory usage may turn out to be a problem. The COLOR_ALLOCATE keyword controls the amount of memory allocated for the color and property arrays. The syntax is:

  ```
  COLOR_ALLOCATE {0|1}
  ```

Using a zero value, the color vector (and possibly the properties vectors, depending on PROP_MODE) is allocated only for the current IWST. This saves memory but makes the computation slower, specially if the GRADIENT_MODE is 2 or 3. In addition, setting COLOR_ALLOCATE to 0 deactivates the passing of colors through the contacts (DOCONTACTS and NOCONTACTS keywords) and the plotting (sets PLOT_MODE to 0). If COLOR_ALLOCATE is 1, the color (and optionally the properties) of all the IWST are saved. By default, COLOR_ALLOCATE is 1 if maxlevel.i <= 8 and 0 if the maximum level is higher.

- In cases where only the properties of some of the atoms are interesting, there exists the possibility of deactivating the integration of some atoms by using the INACTIVE and ACTIVE keywords. Tetrahedra belonging to an inactive atom are not integrated (apart from the volume, see above). The termini of inactive atoms are all assigned to a common 'unknown' value (the same value as the one assigned when a gradient path is 'killed' in non-periodic cells, see KILLEXT above). If the number of inactive atoms is large, the 'painting' of unknown termini becomes more frequent and applies to more atoms, thus saving time even in the termini localization step. The ACTIVE and INACTIVE keywords are used:

  ```
  INACTIVE 2    !  deactivate integration of atom 2
  INACTIVE 0    !  deactivate all atoms
  ACTIVE 1      !  activate atom 1
  ```

  By default, all atoms are active.

QTREE is described in J. Comput. Chem. 32 (2010) 291-305. Please, cite this reference if you use this keyword in your work.

12.5 Yu and Trinkle (YT)

```
YT [NNM] [NOATOMS] [ESCHER] [FIELDLIST]
```

The Yu-Trinkle (YT) algorithm uses the reference field to calculate the attraction basins. The reference field must be defined on a grid. Hence it won't work directly with wien2k, elk or aiPI densities, but those can be transformed into a grid by appropriate use of the LOAD keyword. The algorithm proceeds by running over grid nodes in decreasing order of density. If a point has no neighboring points with higher density, then it's a local maximum. If it does, but all of them belong to the same basin then it belongs to the interior of that basin as well. Otherwise, it is sitting on top of the interatomic surface. The actual fraction of a grid point on a IAS belonging to a particular basin is calculated by evaluating the trajectory flow to neighboring points.

The YT algorithm is described in J. Chem. Phys. 134 (2011) 064111 which should be consulted for further details. Please, cite this reference if you use this keyword in your work.

The located maxima are identified by default with the closest nucleus. If non-nuclear maxima are expected, use the NNM keyword to assign only maxima that are only within 1 bohr of the closest atom. The NOATOMS option is appropriate for scalar fields where the maxima are not expected to be at the atomic positions (or at least not all of them). If NOATOMS is used, all the maxima found are given as NNM. This is useful for fields such as the ELF, the Laplacian,...

With the ESCHER keyword, it is possible to write a .m octave script containing the geometry that can be interpreted by escher (run by the escherplot.m file in the src/ subdirectory). With FIELDLIST, critic2 writes a list of all the maxima found and the value of all integrable fields at those points.

Not all the properties defined by the INTEGRABLE keyword are integrated. Only the subset of those properties that are grids, have F or FVAL as the integrand and are congruent with the reference grid are considered. This limitation can be circumvented by using LOAD AS. In addition, no core is used even if the CORE keyword is active. The volume is always integrated. A xyz file (<root>_yt.xyz) is always written, containing the unit cell description (with border, see WRITE) and the position of the maxima, labeled as XX.

Note that in the output ('List of basins and local properties'), 'Charge' refers not to the integrated electron density (because critic2 doesn't know what is an electron density and what not) but to the value of the integral of the reference field in its own basins (which may not make much sense if you are integrating, for instance, the ELF or the Laplacian). Loading a second field and using INTEGRABLE and the field number is the way to go in such cases.

Usage of the YT algorithm for grid integration is strongly recommended, as it is much more efficient, robust and accurate than the other alternatives combined with spline interpolation. BADER is, however, more memory-efficient than YT, so it is recommended for very large grids instead.

12.6 Henkelman et al. method (BADER)

The algorithm by Henkelman et al. uses the BADER keyword:

```
    BADER [NNM] [NOATOMS] [ESCHER] [FIELDLIST]
```

The BADER algorithm uses the reference field to calculate the basins; this field must be defined on a grid. BADER assigns grid nodes to basins using the near-grid method incrementally described in Comput. Mater. Sci. 36, 254-360 (2006), J. Comput. Chem. 28, 899-908 (2007), and J. Phys.: Condens. Matter 21, 084204 (2009). Please, cite these references if you use this method.

The output and the option keywords have the same meaning as YT. Using BADER as an alternative to YT is recommended in very large grids because its more efficient memory usage.

13 Non-covalent interaction visualization (NCIPLOT)

```
NCIPLOT
  ONAME root.s
  CUTOFFS rhocut.r dimcut.r
  RHOPARAM rhoparam.r
  RHOPARAM2 rhoparam2.r
  CUTPLOT rhoplot.r dimplot.r
  VOID void.r
  RTHRES rthres.r
  INCREMENTS x.r y.r z.r
  NSTEP nx.i ny.i nz.i
  ONLYNEG
  NOCHK
  CUBE x0.r y0.r z0.r x1.r y1.r z1.r
  CUBE file1.xyz file2.xyz ...
  MOLMOTIF
  FRAGMENT file.xyz
  FRAGMENT
   x.r y.r z.r # (in angstrom!)
   ...
  ENDFRAGMENT
ENDNCIPLOT
```

Calculates the density and reduced density gradient on a cube for the visualization of non-covalent interactions. The output files are

- <root>.dat: a 2-column file with reduced density gradient (col. 2) vs. density (col. 1) calculated at the points of a grid. The density in column one is multiplied by the sign of the second eigenvalue of the density Hessian.
- <root>-dens.cube: a cube containing the electron density times the sign of the second Hessian eigenvalue times 100.
- <root>-grad.cube: a cube containing the reduced density gradient (RDG).
- <root>.vmd: the VMD script for convenient visualization of the results.

The root of the files can be changed using the ONAME keyword (default: <root>).

By default, the region represented is the whole unit cell, with step lengths of 0.1 bohr in each direction. The step lengths or the number of points in each axis can be controlled with the keywords:

INCREMENTS `x.r y.r z.r`

x.r, y.r and z.r are the step lengths in each direction.

NSTEP `nx.i ny.i nz.i`

NSTEP defines the number of steps in each direction.

If plotting the whole unit cell is not convenient, a parallelepipedic region can be extruded from the crystal with the keyword:

CUBE `x0.r y0.r z0.r x1.r y1.r z1.r`

where (x0.r y0.r z0.r) and (x1.r y1.r z1.r) are the cube limits in crystallographic coordinates. A region (much) larger than one periodic cell can also be selected with this method. In addition, it is possible to define the cube containing a fragment of the crystal using:

CUBE `file1.xyz file2.xyz ...`

Define the limits of the cube using the crystal fragments contained in file1.xyz, file2.xyz,... A small border (RTHRES) is added around the region exactly containing all the atoms. The xyz files have the usual xyz format (in angstrom), in the same spirit as the FRAGMENT keyword. The xyz, however, are not used as fragments for the NCI calculation.

Some cutoffs are relevant to the visualization of the NCI:

CUTOFFS `rhocut.r dimcut.r`

These cutoffs apply to the writing of density and reduced density gradients (respectively) to the dat file. If, at a given point, the density is above rhocut.r or the reduced density gradient is above dimcut.r, then the point is not written to the dat file. Defaults: rhocut.r = 0.2 and dimcut.r = 2.0 for loaded densities and 1.0 for promolecular densities.

CUTPLOT `rhoplot.r dimplot.r`

When the density is greater than rhoplot.r, the rdg in the -grad.cube file is set to 100d0, effectively eliminating the point from the isosurface plot. Also, the color scale represented in RDG isosurfaces ranges from -rhoplot.r to rhoplot.r. The default value is 0.05 (selfconsistent densities) or 0.07 (promolecular). The dimplot.r controls the isosurface value to be represented in VMD. Default: 0.5 (SC) or 0.3 (promolecular).

MOLMOTIF

Complete the molecules that lie across unit cell faces by using atoms in the neighboring cells.

ONLYNEG

Represent only the points where the second eigenvalue of the Hessian is negative.

```
NOCHK
```

Do not read or write the checkpoint file.

```
VOID void.r
```

Represent only the points where the promolecular density is lower than void.r.

```
FRAGMENT file.xyz
FRAGMENT
 x.r y.r z.r # (in angstrom!)
 ...
ENDFRAGMENT
```

In the current version of NCIplot it is possible to define molecular fragments in order to focus on some part of the crystal, or some particular interaction. This is done by using the FRAGMENT environments. Each FRAGMENT block defines one fragment and only the intermolecular interactions between fragments are represented (hence, you need at least two blocks). The atomic positions (in Cartesian coordinates, the units are angstrom) of the atoms in the fragment appear inside. To obtain the list of atoms, the recommended procedure is to write an xyz file (using the WRITE keyword), then cutting it into pieces (using, for instance, avogadro) and then placing the resulting atom lists in FRAGMENT environments.

Alternatively, the fragment can be read from an xyz file.

There are three options that control the behavior of the fragments: RTHRES, RHOPARAM and RHOPARAM2.

```
RTHRES rthres.r
```

When fragments are used, the density and rdg grids are reduced to a piece encompassing the fragments, with a border of rthres.r bohr (default: 2.0).

```
RHOPARAM rhoparam.r
```

Consider only the points where none of the fragments contributes more than rhoparam.r times the total promolecular density (default: 0.95).

```
RHOPARAM2 rhoparam2.r
```

Consider only the points where the sum of the density of all fragments is more than rhoparam2.r times the total promolecular density (default: 0.75). Note that the fragments need not include all atoms in the crystal.

Some advice regarding the execution of NCIPLOT:

- If the density is given on a grid, it is usually much faster and the result is much better if core-augmentation (ZPSP) is *not* used. The reason is that if the core augmentation is not present, the reduced density gradient and the Hessian components can be calculated by Fourier transform (which is smoother) and it is not necessary to sum over neighboring atoms. The NCI regions are in the interstitial, where the density is mostly correct anyway, and the plots are only qualitative.

- Likewise, any option that activates the calculation of the promolecular density (VOID or FRAGMENT) is going to be expensive because it involves a sum over neighboring atoms. In those cases, it is recommended to calculate the promolecular density once for a certain grid, then use the checkpoint file.
- Some programs (most notably VMD) have problems dealing with non-orthogonal cells. There's little critic2 can do about this, as the cube files on output are correctly written. Using FRAGMENT or CUBE in critic2 so that a orthogonal piece of the crystal is represented may help.

14 Miscellaneous commands

14.1 Powder diffraction (POWDER)

```
POWDER [TH2INI t2i.r] [TH2END t2e.r] [{L|LAMBDA} l.r]
       [FPOL fpol.r] [NPTS npts.i] [SIGMA sigma.r]
       [ROOT root.s]
```

Generate the powder diffraction pattern for the current crystal structure. Consider only the 2*theta range going from t2i.r (default: 5 degrees) to t2e.r (def: 90). The wavelength of the incident radiation is given by l.r (in angstrom). The polarization of the x-ray radiation affects the treatment of the resulting intensities. The default is fpol.r = 0, corresponding to unpolarized light. For synchrotron radiation, use fpol.r = 0.95. npts.i is the number of points in the generated spectrum. Gaussian broadening is used on the observed peaks, with width parameter sigma.r (def: 0.05). By default, two files are generated: <root>_xrd.dat, containing the 2*theta versus intensity data, and <root>_xrd.gnu, the gnuplot script to plot it. The name of these files can be changed using the ROOT keyword.

14.2 Commands for crystallographic computation

```
NEWCELL x1.r y1.r z1.r x2.r y2.r z2.r x3.r y3.r z3.r
        [INV|INVERSE]
```

Transform the crystal structure description by using a new unit cell given by the vectors x1, x2, x3 (in crystallographic coordinates) relative to the old unit cell. The x1, x2, x3 vectors must be pure translations of the old cell; either lattice vectors, centering vectors, or combinations of the two. NEWCELL unloads all fields (except the promolecular density) and clears the critical point list. If the INV (or INVERSE) keyword is used, the input vectors correspond to the crystallographic coordinates of the old cell in the new coordinate system.

The NEWCELL keyword is useful for building supercells or for performing routine but tedious crystallographic transformations. For instance, given a face-centered cubic lattice and the conventional cubic cell one can find the primitive (rhombohedral) cell by doing:

```
CRYSTAL LIBRARY mgo
NEWCELL 1/2 1/2 0 1/2 0 1/2 0 1/2 1/2
```

Likewise, if the current cell is rhombohedral, the same NEWCELL order but including the INVERSE keyword transforms to the cubic. That is:

```
CRYSTAL LIBRARY mgo
NEWCELL 1/2 1/2 0 1/2 0 1/2 0 1/2 1/2
NEWCELL 1/2 1/2 0 1/2 0 1/2 0 1/2 1/2 INVERSE
```

gives a unit cell and crystal structure description that is equivalent to the initial one read from the library.

```
WS
```

Find the Wigner-Seitz cell of the system and print information about its partition in tetrahedra. Useful for debugging.

```
ENVIRON [SHELLS nshel.i] [POINT x0.r y0.r z0.r]
```

Print the shells of atomic neighbors for every non-equivalent atom in the cell, up to nshel.i shells (default: 10). If a point is given with the POINT keyword (crystallographic coordinates), then print the atomic environment of that point instead.

```
PACKING [VDW] [PREC prec.r]
```

Compute the packing ratio assuming atomic spheres with radius equal to the nearest neighbor distance divided by 2. If VDW is used, then use the van der Waals radii and allow the spheres to overlap. This option is currently implemented by building a grid on the unit cell and checking whether its points are inside any atomic sphere, which is not very efficient. The PREC allows controlling the precision of the packing ratio calculated using the VDW keyword. If PREC is used, expect an error in the percent packing ratio in the order of prec.r. The default prec.r is 0.1.

```
IDENTIFY [ANG|ANGSTROM|BOHR|AU|CRYST|RECIPROCAL]
 x.r y.r z.r [ANG|ANGSTROM|BOHR|AU|CRYST|RECIPROCAL]
 ...
 file.xyz
ENDIDENTIFY
```

Identify the coordinates in the input and match them against the list of atoms and critical points. If a coordinate is close (1e-4) to an atom or CP, the corresponding indices as well as the crystallographic coordinates are written to the output. The input can come as either the coordinates of the points themselves or a filename pointing to an xyz file.

The default units are crystallographic coordinates, but they can be modified with one of the keywords that follow IDENTIFY. For specific points, the unit can be changed by specifying a keyword after the three coordinates. The units in the xyz file are angstrom (the xyz file has to have the usual syntax, with the number of atoms in the first line and the title in the second line).

In addition, critic2 provides the vertices of the cube that encompasses all the points in the list that did match an atom or CP, and the same cube with a 2 bohr border.

The RECIPROCAL keyword is special in that points under this keyword are assumed to belong to reciprocal space. Critic2 will calculate the multiplicity using the reciprocal point group operations, which can be useful for determining k-point weights.

```
EWALD
```

Calculate the electrostatic energy of the lattice of point charges using Ewald's method. The atomic charges are defined using the Q keyword.

14.3 Hirshfeld charges

```
HIRSHFELD
```

Calculate the Hirshfeld charges. Only for fields on a grid.

14.4 Scanning Tunneling Microscope (STM) plots

```
STM [CURRENT [curr.r]|HEIGHT [hei.r]] [TOP top.r]
    [{CELL|CELLS} nx.i ny.i] [NPTS n1.i n2.i]
    [LINE x0.r y0.r x1.r y1.r npts.i]
```

Generate an STM plot for the current system, comparable to those obtained in actual STM experiments. The STM keyword should be applied only to systems meeting the following requirements:

- The geoemtry is a slab, possibly with molecules adsorbed to one or both faces.
- The slab and the corresponding vacuum are perpendicular to one of the crystallographic axes (a, b, or c). The two angles related to the perpendicular axis must be right. For instance, a typical situation is one where a and b form a hexagonal cell in the plane, and c is perpendicular to the slab. The angles related to c (alpha and beta) both are 90 degrees.

- The slab has two faces: one with vacuum above and one below. Only the face with vacuum above it will be used.

- The STM plot calculation uses the reference field. For results consistent with the Tersoff-Hamann approximation, the reference field should be the local density of states (LDOS, in the following) at the Fermi level, that is, the density generated by the Kohn-Sham states close to Ef. This field can be obtained easily from common condensed-matter electronic structure programs. For instance, in QE, one would use plot_num 3 or 5 in pp.x.

There are two modes of operation in the STM keyword. In the default mode (CURRENT), the plot gives the height (in angstrom) for which the LDOS equals curr.r (a.u. units, default: 1d-3). This corresponds to the constant-current mode in STM experiments. The other mode is HEIGHT, in which the LDOS at a constant height over the surface is plotted. This is the constant-height mode in STM experiments. The hei.r option to HEIGHT is the fractional coordinate along the perpendicular axis of the plot plane. By default, critic2 uses the fractional coordinate corresponding to the last atom before the vacuum.

The CURRENT plot shows the distance in angstrom with respect to a point that also corresponds to the last atom before the vacuum. In the case of a surface with adsorbed molecules on it, this is not very useful; usually one wants to refer this value to the top of the slab surface. The TOP keyword allows changing this value. top.r is the fractional coordinate along the perpendicular axis that will be used as reference for the CURRENT plot.

The CELL (or CELLS) keyword controls the number of unit cells plotted in each in-plane crystallographic direction (default: 1 1). The number of points plotted in each in-plane unit cell is given by the NPTS keyword (n1.i and n2.i). By default, critic2 uses 51x51 points or, if the reference field is a grid, the number of grid points in each of those directions. If a grid is used, letting critic2 use the grid geometry is strongly recommended because it is more time-efficient and accurate.

On output, two files will be written. The <root>_stm.dat file contains the 2D data for the STM signal on the selected plane. The <root>_stm.gnu file is an example script that generates a plot similar to those found in the literature.

The LINE keyword makes critic2 plot a line instead of a plane. The line goes from (x0,y0) to (x1,y1), in fractional coordinates for the in-plane crystallographic axes. npts.i points along the line are calculated. The output file is <root>_stm_line.dat.

The current implementation of STM has benefited from the code and the guidance kindly provided by Enrico Benassi (see THANKS).

14.5 The exchange-hole dipole moment (XDM) model of dispersion

The XDM module calculates the dispersion energy using the exchange-hole dipole moment (XDM) model. See J. Chem. Phys. 127, 154108 (2007), J. Chem. Phys. 136, 174109 (2012), and J. Chem. Phys. 138, 204109 (2013) for more details.

```
XDM [RHO irho.i] [TAU itau.i] [ELF ielf.i] [PDENS ipdens.i]
    [CORE icor.i] [LAP ilap.i] [GRAD igrad.i] [RHOAE irhoae.i]
    [XB ib.i] [XA1 a1.r] [XA2 a2.r] [ONLYC] [UPTO {6|8|10}]
```

XDM uses the electron density (RHO), the kinetic energy density (TAU), the Laplacian (LAP), and the gradient of the electron density (GRAD) to compute the exchange-hole dipole moment in the Becke-Roussel model (B). The promolecular density (PDENS) and the core density (CORE) are used to calculate a Hirshfeld partitioning of the unit cell. All of these fields have to be available or are calculated when running XDM. The corresponding keywords accept an integer, corresponding to a previously LOADed field. During the XDM run, cubes for all of these fields are generated so they can be loaded in subsequent runs. Note that the PDENS cube is the same as generated in NCIPLOT (except for changes in size).

The list of requierements is:

- RHO: the electron density. By default, irho.i is the reference field. This field is required in XDM. It is also required to give the pseudopotential charges using the ZPSP keyword for all types of atoms in the system.
- TAU: the kinetic energy density. It is used in the calculation of B, and can be extracted from the ELF. Hence, it is required except if the ELF or the B is given. If the ELF is used instead of TAU, a cube file (-tau.cube) is written.
- ELF: the electron localization function. Can be used in place of TAU. This is useful because most programs (e.g. QE, VASP) generate cubes for the ELF but not for the kinetic energy density.
- PDENS: the promolecular density. It is generated by critic2 if not present in the XDM call, and written to a cube file (-pdens.cube) for future use.
- CORE: the core density. It is generated by critic2 if not present in the XDM call and written to a cube (-core.cube), unless the B and RHOAE are given, in which case it is ignored. The ZPSP of all atoms is required in order to calculate this quantity.
- LAP: the Laplacian of the electron density. It is generated by Fourier transform of RHO unless it is given or B is given, in which case it is not needed and its calculation is skipped.

- GRAD: the gradient of the electron density. It is generated from RHO unless B is given. If B is available, the calculation of GRAD is skipped.
- RHOAE: the all-electron density on a cube. If given, replaces the pseudo-density plus core in the calculation of the atomic volumes.
- XB: the exchange-hole dipole moment in the Becke-Roussel model. Calculated from the above unless given.

The usual way of running XDM for the first time is:

```
CRYSTAL rho.cube
ZPSP C 4 O 6 H 1
LOAD rho.cube
LOAD elf.cube
XDM elf 2
```

This generates several cube files: root-tau.cube, root-pdens.cube, root-core.cube, root-lap.cube, root-grad.cube, and root-b.cube. Subsequent runs can circumvent the calculation of B, PDENS, and CORE by doing:

```
CRYSTAL rho.cube
ZPSP C 4 O 6 H 1
LOAD rho.cube
LOAD root-b.cube
LOAD root-pdens.cube
LOAD root-core.cube
XDM XB 2 PDENS 3 CORE 4
```

Note that passing RHO 1 is not necessary because rho.cube is the first field loaded (hence the reference) and assumed by default to be the density by XDM. Again: the ZPSP of all atoms are needed for an XDM calculation. Also, only closed-shell (non-spinpolarized) systems can be calculated for now. During the calculation, cubes for almost all the properties above are generated so they can be reused in future calculations. The other options are:

```
XA1 [a1.r]
```

The value of the a1 damping parameter (adimensional). Default: 0.6836 (PW86PBE parametrization for QE).

```
XA2 [a2.r]
```

The value of the a1 damping parameter (in angstrom). Default: 1.5045 (PW86PBE parametrization for QE).

```
ONLYC
```

Calculate the dispersion coefficients but not the dispersion energy, forces, and stress. By default, they are calculated.

```
UPTO {6|8|10}
```

Only calculate the contributions to the energy coming from the C6 term (6), from the C6 and C8 terms (8) and from C6, C8, and C10 (10). The latter is the default.

14.6 Control commands and options

```
ODE_MODE {EULER|RKCK|DP} [step.r [gradeps.r]]
```

Select the gradient path integration algorithm. Euler is plain explicit Euler. RKCK, Runge-Kutta embedded 4-5th order method (Cash-Karp parametrization). DP, Dormand-Prince 4-5th order. step.r is the initial step size and gradeps.r is the gradient norm termination criterion for the gradient path. This keyword applies to all gradient paths except those in qtree.

```
INT_RADIAL [TYPE {GAULEG|QAGS|QNG|QAG}] [NR nr.r]
           [ABSERR aerr.r] [RELERR rerr.r]
           [ERRPROP prop.i]
           [PREC delta.r]
```

Choose the radial integration method (e.g. inside spheres or basins). The TYPE keyword selects the qudrature method:

- GAULEG: Gauss-Legendre.
- QAGS: quadpack's dqags (general-purpose, extrapolation, globally adaptive, end-point singularities). All Q methods are sometimes unstable for heavy atoms and big beta-spheres, but this does not happen very often.
- QNG: quadpack's dqng (smooth integrand, non-adaptive, Gauss-Kronrod(Patterson))
- QAG: quadpack's dqag (general-purpose, integrand examiner, globally adaptive, Gauss-Kronrod)

The number of radial integration points, if appropriate (GAULEG,QAG) is selected with NR. If the selected method is QAG, the number of points may vary from nr.r. The allowed intervals are: 7 - 15, 10 - 21, 15 - 31, 20 - 41, 25 - 51, 30 - 61. Critic2 selects the appropriate interval by comparing the given nr.r to the lower limits of these intervals.

ABSERR is the requested absolute error for QUADPACK integrators. RELERR is the requested relative error for QUADPACK integrators. ERRPROP controls the property for which the error is estimated. If prop.i corresponds to one of the integrable properties then RELERR and ABSERR apply only to it. The selected property guides the adaptive integration procedure. If prop.i does not represent one of the integrable properties, the maximum of the absolute value of the properties vector is used. Note : the option where max(abs(prop)) is used is unstable. Some spheres (usually associated to heavy atoms) may integrate to nonsense, depending on the optimization levels of the compiler. Therefore, I have disabled it by default.

In the case of basin integrations, PREC controls the precision in the determination of the interatomic surface.

Default: GAULEG, nr.r = 50, aerr.r = 1d0, rerr.r = 1d-12. The default errprop is the field value for fields on a grid and the Laplacian for the rest.

```
ROOT root.s
```

Change the <root>. The root is used as prefix for most of the auxiliary files written by critic2.

```
SUM id.i
MAX id.i
MIN id.i
MEAN id.i
COUNT id.i eps.r
```

Using the field on a grid id.i, calculate the sum of the node values (SUM), the maximum value (MAX), the minimum (MIN), the average (MEAN), or count the number of elements that are greater than eps.r (COUNT).

```
BENCHMARK [nn.i]
```

Benchmark the speed of the field calculation using nn.i points (default: 10000)

```
{RUN|SYSTEM} command.s
```

Execute a shell command.

```
ECHO string.s
```

Write the string to the output. Useful for partitioning long outputs.

```
expression.s
```

If the input is not identified as any of the keywords above, then evaluate the line as an arithmetic expression. Useful for simple calculations in the command line (with critic2 -q).

```
END
```

Terminates the execution.

15 Appendix

15.1 Related programs and output file formats

Some programs required or just useful when working with critic2 are:

- **gnuplot**: most of the output critic2 creates are text files with column organized grid data, readily formatted to use with gnuplot. Also, in some tasks, critic2 also generates the gnuplot scripts required for the representation.
- **geomview**: some atomic basin plots are created in geomview's OFF and COFF format, which are described below.
- **tessel**: tessel is a generalized plotting program for crystal structures. Critic2 provides input files for tessel (mainly in FLUXPRINT). Also, BASIN and DBASIN files, containing the information about the atomic basins are generated, which are read and plotted by tessel.
- **avogadro**: can be used to manipulate xyz files for nciplot.

The description of the output file formats in critic2 follows.

INT files

The INT files contain a description of an IAS surface for a number of rays. The rays are given by a fixed 2d angular quadrature, specified in the first line of the file. The successive lines are the r_ias, in the precise order needed. Only for internal use of CRITIC.

BASIN files

The surface of an atomic basin is approximated in critic2 by a polyhedron, with vertices on the rays on which the zero-flux surface has been determined. This surface, named basin from now on, is approximately described as a number of polygonal facets, each being a ordered list of vertices.

Let us consider, in addition, several scalar properties evaluated on the vertices of the surface. We can select any of this properties to create a color map of the scalar property on the basin surface.

This files can only be used to analyze those surfaces that are monovalued in spherical coordinates (i.e. surfaces that have one and only one value of the radial coordinate for every angular point).

The structure of the BASIN files is:

- **Rec. 0** : "#" comment.

  Comment lines starting with "#" may appear anywhere.

- **Rec. 1** : nvert, nface, nedge

  Number of vertices, faces and edges of the polyhedron.

- **Rec. 2**: npropty

  Number of properties (scalars or scalar components) that will be given for each point.

- **Rec. 3** : (propname(i), i = 1, npropty)

  Alphanumerical label for each property. Blank characters (one or more) separate the properties. Notice that vectors are given as three separate components, etc.

- **Rec. 4i, i = 0, nvert-1** : x(i), y(i), z(i), (prop(j,i),j=1,npropty)

  For each vertex: the Cartesian coordinates and the values of all the properties given for this point.

- **Rec. 5j, j = 0, nface-1** : nv, (ivert(k,j), k=1,nv)

  The number of vertices of this face is nv (nv=3 in the triangular tesselations and 4 in the quadrilateral ones). For each vertex in this face, the number of order in the previous vertex list. Remember that vertices are numbered from 0 to nvert-1.

DBASIN files

The DBASIN files contain the description of the basin of a point, and the value of a scalar property inside the basin (e.g. the electron density). Those data are used to plot the surfaces of constant value of the scalar property.

A regular grid of NPOINT points is defined along each ray from the origin (excluded) to the limit of the basin for this ray. The scalar property is computed in these points. Notice that the grid is, in principle, different for each ray.

The structure of a DBASIN file is:

- **Rec. 0** : "#" comment

  Comment lines starting with "#" may appear anywhere.

- **Rec. 1** : nvert, nface, nedge

  Number of vertices, faces and edges of the polyhedron.

- **Rec. 2** : npoint, xnuc, ynuc, znuc, rhonuc

  Number of sampled points along each ray. Cartesian (x,y,z) position of the nucleus. Electron density at the nucleus.

- **Rec. 3i, i = 0, nvert-1** : x(i), y(i), z(i), (rho(j,i),j=1,npoint)

  For each vertex: the Cartesian coordinates and the values of the electron density at the grid points.

- **Rec. 4j, j = 0, nface-1**: nv, (ivert(k,j), k=1,nv)

  The number of vertices of this face is nv (nv=3 in the triangular tesselations and 4 in the quadrilateral ones). For each vertex in this face, the number of order in the previous vertex list. Remember that vertices are numbered from 0 to nvert-1.

OFF and COFF files

The OFF/COFF files can be viewed and printed with geomview. The structure is:

- **Rec. 0** : "#" comment.

  Comment lines starting with "#" may appear anywhere.

- **Rec. 1** : file_type

  Either OFF or COFF keyword.

- **Rec. 2** : nvert, nface, nedge

  Number of vertices, faces and edges of the polyhedron.

- **Rec. 3** : (x(i),y(i),z(i), [r(i),g(i),b(i),alpha(i)],i = 0, nvert-1)

  Cartesian coordinates (x,y,z) of the vertices. In the case of COFF files, additional information is given regarding the RGB color associated to the vertex, and the degree of transparency (alpha) of the surface at this position.

- **Rec. 4j, j = 0, nface-1** : nv, (ivert(k,j), k=1,nv)

  The number of vertices of this face is nv (nv=3 in the triangular tesselations and 4 in the quadrilateral ones). For each vertex in this face, the number of order in the previous vertex list. Remember that vertices are numbered from 0 to nvert-1.

16 Copyright notice